**Life Documentation**

**Overview**

Life application provides users with a comprehensive platform for managing fitness-related activities, tracking body measurements, creating meal and workout plans, and collaborating with personal trainers. This document outlines the functional and technical aspects of the application to facilitate ongoing development, maintenance, and support.

**Purpose**

The documentation serves as a guide for developers and functional teams to understand the application's architecture, workflows, and technical details. It ensures continuity in development and support, enabling efficient onboarding and troubleshooting.

**Target Audience**

* **Developers**: To continue development or fix issues.
* **Functional Teams**: To assist in functional testing and process understanding.
* **Support Teams**: To address user queries and issues.

**Features**

1. **Current Body Measurements**:
   * Tracks weight and body measurements including Height, Weight, Neck, Shoulder, Chest, Arm, Forearm, Waist, Hips, Thigh, and Claves (front, side, and back images).
   * Logs data per date.
2. **Target Measurements**:
   * Allows users to set target body measurements including Height, Weight, Neck, Shoulder, Chest, Arm, Forearm, Waist, Hips, Thigh, and Claves.
   * Target Measurements are logged per date.
3. **Calculators**:
   * Includes BMI, BMR, Body fat, TDEE, and Macro’s calculators. The equations were referenced to online platforms
   * Users can log results to facilitate tracking, or perform quick calculations without saving.
4. **Workouts**:
   * Users can create training plans with 500+ predefined exercises (videos + descriptions). Each plan can have a specific timeframe or can be unlimited. Time frames ensure that the user switch between the plans as it is recommended to alternate between plans every now and then.
   * You can add multiple workouts in a single plan, workouts can be per days of the week, muscle group, etc. The cycle repeats itself when all workouts in the plan has been accomplished.
   * For each workout, you can “Skip”, “Update”, “Restart”, “Remove” or “Complete.
     + Skip: is used to skip the workout in the current cycle. This workout will be reactivated in the next cycle upon completion of the current one. If the workout has been started, and this function is used, the data that has been input in the workout will not be saved to the database.
     + Update: if you need to add, remove, edit number of sets in the workout, you can use this functionality.
     + Restart: if you have performed a specific workout, and you want to redo it within the same cycle, use this functionality.
     + Remove: this will remove the workout from the cycle.
     + Complete: when starting a workout, and for any reason was not able to finish it, this functionality can be used to save the performed exercises and skip the remaining ones.
   * Each workout specifies target muscle group with up to 3 major and minor muscle activations, along with the type of exercise (compound, isolation, or stability) and equipment and weights being used.
   * Users can add custom exercises, but only allowed to upload images.
   * Search functionality across the exercises by: “Our workouts”, “Your workouts”, “Equipment” used, and “Muscles” group.
   * Tracks sets, weights, and reps.
   * Timer between sets can be customized from “App Settings”, where you can assign different rest timers based on the exercise type (compound, isolation, or stability).
   * At any given time, you can use the “Open Calendar” to check the performed workout in a specific date.
5. **Meal Plan**:
   * Integrates calculated macros into a daily meal plan.
   * Provides graphs comparing required and consumed macros.
   * Consumed meals are restarted daily at 12 am.
   * Includes a predefined list of 1800 foods, foods are with AI generated images; users can add custom meals. Based on the type of food, default image is chosen.
6. **Personal Trainer Search**:
   * Lists trainers with “About me” section, country, qualifications, achievements, availability and bundles.
   * Users can filter based on country, rating, gender, price and discounts.
   * Users can respond to personal trainer questionnaire, and subscribe using payment gateway based on the country of the user and the bundle created by the trainer from the pricing page.
   * Currently, all payments are received in EGP based on Egypt’s policies. PayPal has been hashed from the code.
7. **Personal Trainer Features**:
   * Sets pricing by country and currency.
   * Can create multiple bundles based on the timeframe the trainer provides the services in.
   * Add discounts, this is shown to the trainees when they search for personal trainers.
   * Check how much he would be receiving after deduction of commissions of payment gateway, platform, and bank transfer fees.
   * Request funds withdrawal by providing bank details. Wallet is reduced as soon as admin processes the payment.
   * Upload own image. (Requires approval from admin to be processed)
   * Create questionnaires that trainees respond to before subscribing.
   * Add certifications, social media links and “About me” section. (Requires approval from admin to be processed)
   * Sets trainee capacity and subscription acceptance toggle.
   * Create predefined workout plans and meal plans that can be easily assigned to the trainees.
   * Predefined meals and workouts can be edited for each trainee independently, or new plans can be created from scratch.
   * Trainer can (within the subscription period) check the starting and current weights, workouts, and meals consumed by the trainees for a given date.
   * Check the status of active and inactive trainees with their corresponding subscription dates.
   * Create and upload workout videos to server and Cloudflare.
8. **Admin Features**:
   * Manages commissions for platform, bank, and payment gateway.
   * Ability to assign specific commission per trainer to motivate them to get more trainees.
   * Handles trainer approval and withdrawal requests.
   * Settle payments against trainers by specifying amount. Settlement happens manually outside the platform.

**Technical Details**

**Architecture**

* **Mobile Application**: Handles local data and real-time user interactions.
* **Backend**: Centralized database and APIs for local and online functionalities.
* **Cloudflare Integration**: Stores and serves multimedia content.
* **Payment Gateway:** Integrates trainee payment to Life’s payment gateway account.

**Technology Stack**

* **Frontend**: React Native, Expo, Javascript.
* **Backend**: Laravel (PHP)
* **Database**: MySQL Online DB, Sqllite Offline DB.

**Backend**

* **API Endpoints**:
  + Authentication: Login, registration, forgot password.
  + Data Sync: Measurements, meal plans, workout plans.
  + Trainer Management: review trainees, and assign plans.
  + Payment Management: Subscription fees, withdrawals.
* **Authentication**: Token-based (Laravel Sanctum).

**Database Schema**

* **Users**: Tracks user details, measurements, and goals.
* **Workouts**: Stores predefined and custom exercises.
* **Meals**: Contains predefined and user-added meals.
* **Subscriptions**: Links trainers and trainees with plan details.
* **Transactions**: Tracks payments and commissions.

**Frontend**

* **Component Management**:
  + Measurements: maintain measurements.
  + Calculators: different calculators including: BMR, BMI, Bodyfat, TDEE, and Macros calculator.
  + App Setting: allow users to change language, backup their data, set their rest timers based on type of exercise, set their height, age, and preferred unit of measure.
  + Workout Planner: Manages training schedules.
  + Meal Planner: Logs and analyzes dietary data.
  + Trainer Search: Filters and displays trainers.
  + Trainee Management: for trainer account only, edit trainer’s profile, and manage pricing and trainees.
  + Admin settings: for admin users only, set commissions for banks, payment gateways, and Life commission.
  + Admin wallet: check liabilities to personal trainers, and post payments.

**Integrations**

* **Cloudflare**: Manages media storage.
* **Payment Gateway**: Processes subscriptions and withdrawals.
* **Shared Host**: Hosts backend and database.
* **Sign-in With Google**

**API Setups**

* Cloudflare: Authentication and file management.
* Payment Gateway: Secure transaction processing.

**Workflows**

1. **User Workflows**:
   * Registration, login, and profile management.
   * Data sync and backup to shared host.
   * Subscription workflows for trainees and trainers.
2. **Email Notifications**:
   * Trainer registration and approval.
   * Fund withdrawal requests.
   * Subscription confirmations.
   * Password reset requests.
   * Admin to trainer funds settlement.
   * Change in application commission to trainer.

**Deployment and Scaling**

* **Host Configuration**:
  + Shared host for backend and database.
  + Cloudflare for media storage.
* **Scaling Strategies**:
  + Database optimization.
  + Implementing caching mechanisms for API responses.

**Troubleshooting**

* **Common Errors**:
  + API Authentication Issues.
  + Database sync failures.
  + Media upload errors to Cloudflare.
* **Logging**: Centralized error logging for backend and mobile app.
* **Debugging Tools**: Postman for API testing, browser dev tools.

**Codebase**

* **Folder Structure**:
  + **Frontend**: Organized by components, services, and assets.
  + **Backend**: Follows MVC pattern.
* **Coding Conventions**:
  + Consistent naming conventions.
  + Thorough documentation for critical modules.

**Version Control**

* **Branching Strategy**:
  + Main branch for production.
  + Feature branches for new functionality.
  + Pull requests with code reviews before merging.

**Future Development**

* **Roadmap**:
  + Implementing chat between trainer and trainee.
  + Implementing “Community” that allows user to post and follow certain users.
  + Activating “Allies” for business partners who wish to join Life.
  + Implementing “Medications” to track medications for the users.
  + Implementing a function for users to compete and check other statuses and lifts.

**References**

* **API Documentation**: Links to Cloudflare and Payment Gateway APIs.
* **Libraries**: List of key libraries used in frontend and backend.

**Important Login Credentials**

* Cloudflare: [Stored securely with admin access only]
* Shared Host: [Stored securely with admin access only]
* Payment Gateway: [Stored securely with admin access only]

**Missing Sections (To Be Filled)**

1. **Detailed API Endpoint Specifications**:
   * Request/response formats.
   * Error handling for each endpoint.
2. **Database Schema Diagrams**:
   * Visual representation of table relationships.
3. **Frontend Component Hierarchy**:
   * Detailed explanation of component interdependencies.
4. **Scaling Details**:
   * Specific tools/services used for database and API scaling.
5. **Critical Code Module Descriptions**:
   * Explain functions of critical backend and frontend modules.
6. **Deployment Steps**:
   * Step-by-step guide for staging and production deployment.
7. **Security Measures**:
   * Authentication protocols.
   * Data encryption methods.
8. **Testing Strategy**:
   * Unit, integration, and end-to-end testing methodologies.
9. **User Roles and Permissions**:
   * Detailed mapping of roles, permissions, and access controls.
10. **Trainer Approval Workflow**:
    * Comprehensive description of how admin reviews and approves trainers.

**------------------------------------------------------------------------------**

**1 - Laravel backend:**

1. **Detailed API Endpoint Specifications :**

**A- Laravel backend Apis**

**A.1 : AdminSettingsAppController :**

**A.1.1 :** **AdminSettingsAppUpdateDataIntoDatabase Function**

**It is the function that responsible for updating the settings of the Admin into the database**

**And its route in Api.php file is /AdminSettingsApp-update-data-into-database**

**It sends the json data {speKey: Value,admCom: Value,AlFoTr: Value,xpyEgCom: Value,xpyForCom: Value } from React native app to the database   
after enter data with success it get back with json success message if the failed it get json error message**

**A.1.2 :** **AdminSettingsAppGetDataFromDatabase Function**

**It is the function that responsible for getting the settings of the Admin from the database**

**And its route in Api.php file is /Admin-Settings-App-Get-Data-From-Database**

**It get the json data{ 'AdminSettingsAppRow'=>$firstRow,'message' => 'Admin Settings App Row added successfully'} from the database in case of success**

**A.2 : AuthController:**

**A.1.1 :** **register Function**

**It is the function that responsible for updating the settings of the Admin into the database**

**And its route in Api.php file is /AdminSettingsApp-update-data-into-database**

**It sends the json data {speKey: Value,admCom: Value,AlFoTr: Value,xpyEgCom: Value,xpyForCom: Value } from React native app to the database   
after enter data with success it get back with json success message if it failed it get json error message**

**A.2 : AuthController:**

**A.1.2 :** **registerWithEmailValidation Function**

**It is the function that responsible on the user info whether he/she trainer or trainee make the necessary validations if not pass it send back to tha app json error meesage if it pass then if he/she trainee it creates its account into database if he/she trainer it sends email to the user to agree that he will create account in our platform its apis route in api.php is Route::post('/register', [AuthController::class, 'registerWithEmailValidation']);**

**A.2 : AuthController:**

**A.1.2 :** **createUserAfterHeAuthenticatedHisEmail Function**

**It is the function that responsible for entering the data of the trainer into the database after he prees on agree button in the email and enter the data into User table and TrainerManageMyProfile if creation happen successfully it show response message that Your Account Created Successfully if error happens it show response error message Failed to Create Your Account . the route of the function in api.php route is   
Route::post('/trainerAllDataToSendToHisEmail-insert-data', [AuthController::class, 'createUserAfterHeAuthenticatedHisEmail'])->name('trainerAllDataToSendToHisEmail.approving');  
  
A.2 : AuthController:**

**A.1.3 :** **signIn Function**

**It is the function that responsible for user login if all the info is correct it send back to the app this json data that used in sign process in the app ['message' => 'Login\_successful', 'token' => $token, 'user' => $user,'tokenRaw'=>$userPerToken,'trainerManageMyProfile'=>$trainerManageMyProfile,**

**'userworkoutsData'=>$userworkoutsData] if the info wrong it send error message** **Invalid\_credentials its route in api.php Route::post('/signin', [AuthController::class, 'signIn']);**

**A.2 : AuthController:**

**A.1.4 :** **createUserAfterHeAuthenticatedHisEmail Function**

**It is the function that responsible for sign out from the app if Route::post('/signout', [AuthController::class, 'signout']);**

**A.2 : AuthController:**

**A.1.5 :** **downloadDataForDatabaseToTheAppFunction**

**It is the function that responsible downloading the data of many pages from backend database into the mobile app if it success it send this data to the app json(['userBeginWorkoutsData'=>$userBeginWorkoutsData,'userBodyStatsAndMeasurementsData'=>$userBodyStatsAndMeasurementsData,'userCalculatorsData'=>$userCalculatorsData,'userGymEquipmentsData'=>$userGymEquipmentsData, 'userListOfFoodsData'=>$userListOfFoodsData, 'userPublicSettingsData'=>$userPublicSettingsData,'userPublicWorkoutsPlansData'=>$userPublicWorkoutsPlansData, 'userPublicWorkoutsPlanDaysData'=>$userPublicWorkoutsPlanDaysData, 'userTargetStatsData'=>$userTargetStatsData, 'userTodayMealsData'=>$userTodayMealsData,'userworkoutsData'=>$userworkoutsData] if it failed it send json message error . its route in the api.php Route::get('downloadDataForDatabaseToTheApp',[AuthController::class, 'downloadDataForDatabaseToTheApp']);**

**A.2 : AuthController:**

**A.1.6 :** **redirectToProvider Function**

**It is the function that responsible for sign in with google it’s part of functions that make the process of signin or signup with google it redirect the user to the page of the google that user can select from his google accounts to make sign in with from the app . Its Route in api.php is Route::get('/auth/{provider}/redirect', [AuthController::class,'redirectToProvider']);**

**A.2 : AuthController:**

**A.1.7 :** **redirectToProvider Function**

**It is the function that responsible for sign in with google it’s part of functions that make the process of signin or signup with google it redirect the user to the app with the account info that he selected to signin/signup into the app . Its Route in api.php is Route::get('/auth/{provider}/callback', [AuthController::class,'handleProviderCallback']);**

**A.2 : AuthController:**

**A.1.8 :** **sendResetLinkEmailFunction**

**It is the function that responsible for sending email to the user with the reset link where the user can reset his password. Its Route in api.php is Route::post('/password/email', [AuthController::class, 'sendResetLinkEmail']);**

**A.2 : BeginWorkoutsController:**

**A.2 :** **insertOrUpdateBeginWorkouts**

**It is the function that responsible for sending synch the user data of start workout data from the app to the database . it get back with json message where for success or failure . Its Route in api.php is Route::post('/beginWorkouts-insert-data', [BeginWorkoutsController::class, 'insertOrUpdateBeginWorkouts']);**

**A.2 : bodyStatsController:**

**A.3 :** **insertBodyStatsAndMeasurements**

**It is the function that responsible for sending synch the user data of Body Stats Measurments data from the app to the database . it get back with json message where for success or failure . Its Route in api.php is Route::post('/BSMeasurments-insert-data', [bodyStatsController::class, 'insertBodyStatsAndMeasurements']);**

**A.2 : calculatorsController:**

**A.4 :** **insertCalculator**

**It is the function that responsible for sending synch the user data of all calculators data from the app to the database . it get back with json message where for success or failure . Its Route in api.php is Route::post('/calculators-insert-data', [calculatorsController::class, 'insertCalculator']);**

**A.2 : gymEquipmentsController:**

**A.4 :** **insertOrUpdateGymEquipments**

**It is the function that responsible for sending synch the user data of GymEquipments data from the app to the database . it get back with json message where for success or failure . Its Route in api.php is Route::post('/gymEquipments-insert-data', [gymEquipmentsController::class, 'insertOrUpdateGymEquipments']);**

**A.2 : ListOfFoodsController:**

**A.5 :** **insertOrUpdate****ListOfFoods**

**It is the function that responsible for sending synch the user data of ListOfFoods data that user create from the app to the database . it get back with json message where for success or failure . Its Route in api.php is Route::post('/gymEquipments-insert-data', [gymEquipmentsController::class, 'insertOrUpdateGymEquipments']);**

**A.2 : PayPalController:**

**A.6.1 :** **paymentsToTrainer**

**It is the function that responsible for pay trainee to the trainer the subscription fees to our in our account in payapl . Its Route in api.php is Route::post('paymentsToTrainer',[PayPalController::class, 'paymentsToTrainer'])->name('paymentsToTrainer');**

**A.2 : PayPalController:**

**A.6.2 :** **paymentsToTrainerSuccess**

**It is the function that responsible for handling success of payment by the paymentsToTrainer function after the success of the payment it record the details of payment into TrainerTraineePayments table and activate the subscription to the trainer into the TrainerSubscriptions table in mySql database . Its Route in api.php is Route::get('paymentsToTrainerSuccess',[PayPalController::class, 'paymentsToTrainerSuccess'])->name('paymentsToTrainerSuccess');**

**A.2 : PayPalController:**

**A.6.3 :** **paymentsToTrainerCancel**

**It is the function that responsible for handling failure of payment by the paymentsToTrainer function. after the failure of the payment it backs with Payment is cancelled message. Its Route in api.php is Route::get('paymentsToTrainerCancel',[PayPalController::class, 'paymentsToTrainerCancel'])->name('paymentsToTrainerCancel');**

**A.2 : PayPalController:**

**A.6.4 :** **getOneTrainerGroubedByCurrencyAndAmount**

**It is the function that responsible for getting all payments that paid to the trainer by the trainee. Its Route in api.php is Route::post('getOneTrainerGroubedByCurrencyAndAmount',[PayPalController::class, 'getOneTrainerGroubedByCurrencyAndAmount']);**

**A.2 : PayPalController:**

**A.6.5 :** **getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit**

**(old version for old table form) It is the function that responsible for getting all payments that paid to the trainer by the trainee where it exclude every payment we pay to the trainer and show the rest of his money that left in our account if the calling of the function is failed it get back with json error message . Its Route in api.php is Route::post('getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit',[PayPalController::class, 'getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit']);**

**A.2 : PayPalController:**

**A.6.6 :** **getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit**

**(new version for new table form) It is the function that responsible for getting all payments that paid to the trainer by the trainee where it exclude every payment we pay to the trainer and show the rest of his money that left in our account if the calling of the function is failed it get back with json error message . Its Route in api.php is Route::post('getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit',[PayPalController::class, 'getOneTrainerGroubedByCurrencyAndAmountWithCreditAndDebit']);**

**A.2 : PayPalController:**

**A.6.7 :** **getTrainersGroubedByCurrencyAndAmount**

**It is the function that responsible for getting all trainers and the payments made to them by the trainees (old version for old table form). Its Route in api.php is Route::post('getTrainersGroubedByCurrencyAndAmount',[PayPalController::class, 'getTrainersGroubedByCurrencyAndAmount']);**

**A.2 : PayPalController:**

**A.6.8 :** **getTrainersGroubedByCurrencyAndAmountWithDebitAndCredit**

**It is the function that responsible for getting all trainers and the payments made to them by the trainees (new version for new table form). Its Route in api.php is Route::post('getTrainersGroubedByCurrencyAndAmount',[PayPalController::class, 'getTrainersGroubedByCurrencyAndAmount']);**

**A.2 : PayPalController:**

**A.6.9 :** **payToTrainerFromOurApp**

**It is the function that responsible for make payement to trainer from our account in paypal to his/her account in paypal using his paypal email. Its Route in api.php is Route::post('paymentsToTrainerFromOurApp',[PayPalController::class, 'payToTrainerFromOurApp'])->name('paymentsToTrainerFromOurApp');**

**A.2 : PayPalController:**

**A.6.10:** **paymentsToTrainerFromOurAppAfterPending**

**It is the function that responsible for make payement to trainer from our account in paypal to his/her account in paypal using his paypal email where if the payement it will take you to page that show payement is successful if not it will show payment is failed maybe its like continue to the direct above function . Its Route in api.php is Route::get('paymentsToTrainerFromOurAppAfterPending',[PayPalController::class, 'paymentsToTrainerFromOurAppAfterPending'])->name('paymentsToTrainerFromOurAppAfterPending');**

**A.2 : PayPalController:**

**A.6.11:** **sendTrainerPaymentsRequest**

**It is the function that responsible sending the trainer to us a request to send money to him/her using his paypal email . Its Route in api.php is Route::post('sendTrainerPaymentsRequest',[PayPalController::class, 'sendTrainerPaymentsRequest']);**

**A.2 : PayPalController:**

**A.6.11:** **sendTrainerPaymentsRequestWithBankOption**

**It is the function that responsible sending the trainer to us a request to send money to him/her using his bank account info. Its Route in api.php is Route::post('/send-Trainer-Payments-Request-With-Bank-Option',[PayPalController::class, 'sendTrainerPaymentsRequestWithBankOption']);**

**A.2 : PredefinedMealsController:**

**A.7.1:** **insertOrUpdatePredefinedMeals**

**It is the function that responsible sending data of the predefined meals from our app to the database.note that predefined meals that’s meals that we create into database as admins not normal users. Its Route in api.php is Route::post('/predefinedMeals-insert-data', [PredefinedMealsController::class, 'insertOrUpdatePredefinedMeals']);**

**A.2 : PublicSettingsController:**

**A.8.1:** **insertOrUpdatePublicSettings**

**It is the function that responsible sending data of the public settings from our app to the database. Its Route in api.php is Route::post('/publicSettings-update-data', [PublicSettingsController::class, 'insertOrUpdatePublicSettings']);**

**A.2 : PublicWorkoutsPlanDays:**

**A.9.1:** **insertOrUpdate****PublicWorkoutsPlanDays**

**It is the function that responsible sending data of the sessions that user made on app’s local database in our app to the online database. Its Route in api.php is Route::post('/publicWorkoutsPlansDays-insert-data', [PublicWorkoutsPlanDaysController::class, 'insertOrUpdatePublicWorkoutsPlanDays']);**

**A.2 : PublicWorkoutsPlansController:**

**A.10.1:** **insertOrUpdatePublicWorkoutsPlanDays**

**It is the function that responsible sending data of the workout plans that user made on app’s local database in our app to the online database. Its Route in api.php is Route::post('/publicWorkoutsPlansDays-insert-data', [PublicWorkoutsPlanDaysController::class, 'insertOrUpdatePublicWorkoutsPlanDays']);**

**A.2 : ratingTrainerController:**

**A.11.1:** **ratingPressing**

**It is the function that responsible for make ratings by users to trainers into online database. Its Route in api.php is Route::post('/trainer-rating-sending', [ratingTrainerController::class, 'ratingPressing']);**

**A.2 : ratingTrainerController:**

**A.11.2:** **getAverageRatingsByTrainer**

**It is the function that responsible for getting ratings of the trainers from online database to our app. Its Route::get('/trainer-rating-numbers', [ratingTrainerController::class, 'getAverageRatingsByTrainer']);**

**A.2 : TargetStatsController:**

**A.11.1:** **insertOrUpdateTargetStats**

**It is the function that responsible for sending target stats data of user from offline sqlite database into online database. Its Route in api.php is Route::post('/tarStat-update-data', [targetStatsController::class, 'insertOrUpdateTargetStats']);**

**A.2 : TodayMealsController:**

**A.12.1:** **insertOrUpdate****TodayMeals**

**It is the function that responsible for sending Today Meals data of user from offline sqlite database into online database. Its Route in api.php is Route::post('/TodayMeals-insert-data', [TodayMealsController::class, 'insertOrUpdateTodayMeals']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.1:** **insertOrUpdateTrainerManageMyProfileWithoutGoingMail**

**It is the function that responsible for adding or updating trainer manage my profile data of the trainer into online database directly. Its Route in api.php is Route::post('/TrainerManageMyProfile-insert-data-without-mailing', [TrainerManageMyProfileController::class, 'insertOrUpdateTrainerManageMyProfileWithoutGoingMail']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.2:** **updateApprovedStatusFieldInTrainerManageMyProfile**

**It is the function that responsible for updating the field of aprstu in the TrainerManageMyProfile table. Its Route in api.php Route::post('/TrainerManageMyProfile-update-Approved-Status', [TrainerManageMyProfileController::class, 'updateApprovedStatusFieldInTrainerManageMyProfile']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.3:** **updateisAcceptToFalseFieldInTrainerManageMyProfile**

**It is the function that responsible for updating the field of aprstu in the TrainerManageMyProfile table to false. Its Route in api.php Route::post('/TrainerManageMyProfile-update-isAccept-To-False', [TrainerManageMyProfileController::class, 'updateisAcceptToFalseFieldInTrainerManageMyProfile']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.4:** **openTrainerRejectForm**

**It is the function that responsible for open trainer reject form. Its Route in api.php Route::post('/TrainerManageMyProfile-trainer-reject-form', [TrainerManageMyProfileController::class, 'openTrainerRejectForm']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.5:** **updateApprovedStatusFieldToRejectInTrainerManageMyProfile**

**It is the function that responsible for update appSta field to reject to reject trainer manage my profile. Its Route in api.php Route::post('/TrainerManageMyProfile-reject-inserting-data', [TrainerManageMyProfileController::class, 'updateApprovedStatusFieldToRejectInTrainerManageMyProfile']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.6:** **updateTrainerSpecificCommissionInTrainerManageMyProfile**

**It is the function that responsible for update trainer spesefic commission fiel trSpCm field in trainer manage my profile table. Its Route in api.php Route::post('/TrainerManageMyProfile-trainer-specific-Commission-inserting-data', [TrainerManageMyProfileController::class, 'updateTrainerSpecificCommissionInTrainerManageMyProfile']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.7:** **sendTMPForAppove**

**It is the function that responsible for sending the trainer the data from trainer manage my profile page into the admin email for approving on it . Its Route in api.php Route::post('/TrainerManageMyProfile-sendTMPForAppove', [TrainerManageMyProfileController::class, 'sendTMPForAppove']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.8:** **insertOrUpdateTrainerManageMyProfile**

**It is the function that responsible for adding or updating trainers data into trainer manage my profile table and send an email to him/her to notify him/her that the profile has been approved successfully . Its Route in api.php Route::post('/TrainerManageMyProfile-insert-data', [TrainerManageMyProfileController::class, 'insertOrUpdateTrainerManageMyProfile'])->name('trainerManageMyProfile.approving');**

**A.2 : TrainerManageMyProfileController:**

**A.13.9:** **getProfileByUserId**

**It is the function that responsible for data of the user from three tables** **AdminSettingsApp,** **TrainerManageMyProfile and TrainerSubscriptions . Its Route in api.php Route::get('/get-profile', [TrainerManageMyProfileController::class, 'getProfileByUserId']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.10:** **getTrainersAllTables**

**It is the function that responsible for data of the user from three tables** **User,** **TrainerManageMyProfile , Rating , TrainerPricingCurrency,** **TrainerSubscriptions and** **TrainerPricingDiscount. Its Route in api.php Route::get('/trainers-search', [TrainerManageMyProfileController::class, 'getTrainersAllTables']);**

**A.2 : TrainerManageMyProfileController:**

**A.13.11:** **updateTrainerQuestionnaireInTrainerManageMyProfile**

**It is the function that responsible for updating or adding questionnaire of the trainer into trainerManageMyProfile. Its Route in api.php Route::post('/TrainerManageMyProfile-trainer-Questionnaire-inserting-data', [TrainerManageMyProfileController::class, 'updateTrainerQuestionnaireInTrainerManageMyProfile']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.1:** **insertTrainerMealsPredefinedPlanDays**

**It is the function that responsible for updating or adding trainers predefined meals into online database. Its Route in api.php Route::post('/trainer-predefined-meal-plan-day-insert', [TrainerMealsPredefinedPlanDaysController::class, 'insertTrainerMealsPredefinedPlanDays']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.2:** **getTrainerMealsPredefinedPlanDays**

**It is the function that responsible for getting trainers plan predefined meals from online database. Its Route in api.php Route::get('/get-trainer-predefined-meal-plan-days', [TrainerMealsPredefinedPlanDaysController::class, 'getTrainerMealsPredefinedPlanDays']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.3:** **deleteTrainerMealsPredefinedPlanDayRow**

**It is the function that responsible for delete trainers plan predefined meals row from online database. Its Route in api.php Route::post('/trainer-Predefined-meal-plan-day-deleting', [TrainerMealsPredefinedPlanDaysController::class, 'deleteTrainerMealsPredefinedPlanDayRow']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.4:** **getTrainerMealsPredefinedPlansWithItsDays**

**It is the function that responsible for getting trainers predefined meals plans and its days from online database. Its Route in api.** **Route::get('/get-trainer-meals-predefined-plans-with-its-days', [TrainerMealsPredefinedPlanDaysController::class, 'getTrainerMealsPredefinedPlansWithItsDays']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.5:** **insertTrainerPredefinedMealPlanDaysNewStylePlus**

**It is the function that responsible for inserting trainers predefined meal plan days (NewStyle) into online database. Its Route in api.** **Route::post('/trainer-Predefined-plan-days-insert-new-style-plus', [TrainerMealsPredefinedPlanDaysController::class, 'insertTrainerPredefinedMealPlanDaysNewStylePlus']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.6:** **removeOnePredefinedMealFromMealPlanDaysNewStyleMinus**

**It is the function that responsible for removing trainers predefined meal plan days (NewStyle) from online database. Its Route in api.** **Route::post('/trainer-Predefined-plan-days-remove-new-style-minus', [TrainerMealsPredefinedPlanDaysController::class, 'removeOnePredefinedMealFromMealPlanDaysNewStyleMinus']);**

**A.2 : TrainerMealsPredefinedPlanDaysController:**

**A.14.7:** **getTrainerPredefinedMealPlanDaysForDayNewStyle**

**It is the function that responsible for getting trainers predefined meal plan days for one day (NewStyle) from online database. Its Route in api.** **Route::get('/get-trainer-Predefined-meal-plan-days-for-day-new-style', [TrainerMealsPredefinedPlanDaysController::class, 'getTrainerPredefinedMealPlanDaysForDayNewStyle']);**

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.8:** **insertTrainerMealsPredefinedPlans**

**It is the function that responsible for inserting trainers predefined meal plans into online database. Its Route in api.** **Route::post('/trainer-meals-predefined-plans-insert', [TrainerMealsPredefinedPlansController::class, 'insertTrainerMealsPredefinedPlans']);**

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.9:** **updateTrainerMealsPredefinedPlans**

**It is the function that responsible for updating trainers predefined meal plans into online database. Its Route in api.** **Route::post('/trainer-meals-predefined-plans-update', [TrainerMealsPredefinedPlansController::class, 'updateTrainerMealsPredefinedPlans']);**

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.10:** **insertUnlimitedTrainerMealsPredefinedPlans**

**It is the function that responsible for inserting trainers unlimited predefined meal plans into online database.**

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.11:** **getTrainerMealsPredefinedPlans**

**It is the function that responsible for getting trainers predefined meal plans from online database.** Its route in the api.php Route::get('/get-meals-trainer-predefined-plans', [TrainerMealsPredefinedPlansController::class, 'getTrainerMealsPredefinedPlans']);

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.12:** **deleteTrainerTraineeMealRow**

**It is the function that responsible for deleting trainers predefined meal plan from online database.** Its route in the api.php Route::post('/trainer-meals-predefined-plans-remove', [TrainerMealsPredefinedPlansController::class, 'deleteTrainerTraineeMealRow']);

**A.2 : TrainerMealsPredefinedPlansController:**

**A.14.13:** **getMealPlanDaysForCurrentDate**

**It is the function that responsible for getting trainers meal plan days from online database.** Its route in the api.php Route::get('/get-trainer-trainee-meal-current-plan-days', [TrainerTraineeMealsPlanDaysController::class, 'getMealPlanDaysForCurrentDate']);

**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.1** **insertTrainerPredefinedPlans**

**It is the function that responsible for inserting trainers predefined workouts plans into online database.** Its route in the api.php Route::post('/trainer-predefined-plans-insert', [TrainerPredefinedPlansController::class, 'insertTrainerPredefinedPlans']);

**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.2** **updateTrainerPredefinedPlans**

**It is the function that responsible for updating trainers predefined workouts plans into online database.** Its route in the api.php Route::post('/trainer-predefined-plans-update', [TrainerPredefinedPlansController::class, 'updateTrainerPredefinedPlans']);  
  
**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.3** **getWeightsFirstAndLastMeasurement**

**It is the function that responsible for getting trainees Weights First And Last Measurements from online database.** Its route in the api.php Route::get('/trainee-plans-measurements', [TrainerTraineeWorkoutsPlansController::class, 'getWeightsFirstAndLastMeasurement']);

**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.4** **getTrainerPredefinedPlans**

**It is the function that responsible for getting Trainer Predefined Plans from online database.** Its route in the api.php Route::get('/get-trainer-predefined-plans', [TrainerPredefinedPlansController::class, 'getTrainerPredefinedPlans']);

**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.5** **deleteTrainerPredefinedPlansRow**

**It is the function that responsible for deleting Trainer Predefined Plan from online database.** Its route in the api.php Route::post('/trainer-predefined-plans-remove', [TrainerPredefinedPlansController::class, 'deleteTrainerPredefinedPlansRow']);

**A.2 : TrainerPredefinedPlansController (Workouts):**

**A.15.5** **getPlanDaysForCurrentDate**

**It is the function that responsible for getting Trainer Predefined Plan days from online database.** Its route in the api.php Route::get('/get-trainer-trainee--current-plan-days', [TrainerTraineeWorkoutsPlansController::class, 'getPlanDaysForCurrentDate']);

**A.2 : TrainerPricingCurrencyController :**

**A.16.1** **insertTrainerPricingCurrencyByPressing**

**It is the function that responsible for inserting Trainer pricing rows into online database.** Its route in the api.php Route::post('/TrainerPricingCurrencyByPressing-insert-data', [TrainerPricingCurrencyController::class, 'insertTrainerPricingCurrencyByPressing']);

**A.2 : TrainerPricingCurrencyController :**

**A.16.2** **UpdateTrainerPricingCurrencyByPressing:**

**It is the function that responsible for updating Trainer pricing rows into online database.** Its route in the api.php Route::post('/TrainerPricingCurrencyByPressing-update-data', [TrainerPricingCurrencyController::class, 'UpdateTrainerPricingCurrencyByPressing']);

**A.2 : TrainerPricingCurrencyController :**

**A.16.3** **deletePriceRow:**

**It is the function that responsible for deletting Trainer pricing rows from online database.** Its route in the api.php Route::post('/trainer-pricing-currency-deleting', [TrainerPricingCurrencyController::class, 'deletePriceRow']);

**A.2 : TrainerPricingDiscountController:**

**A.17.1** **insertOrUpdateTrainerPricingDiscountByPressing:**

**It is the function that responsible for inserting Trainer discount rows into online database.** Its route in the api.php Route::post('/TrainerDiscountCurrencyByPressing-insert-data', [TrainerPricingDiscountController::class, 'insertOrUpdateTrainerPricingDiscountByPressing']);

**A.2 : TrainerPricingDiscountController:**

**A.17.2** **getDiscountCurrencyByUserId:**

**It is the function that responsible for getting Trainer discount rows from online database.** Its route in the api.php Route::get('/get-trainer-discount', [TrainerPricingDiscountController::class, 'getDiscountCurrencyByUserId']);

**A.2 : TrainerPricingDiscountController:**

**A.17.3** **deleteDiscountRow:**

**It is the function that responsible for deleting Trainer discount rows from online database.** Its route in the api.php Route::post('/trainer-discount-currency-deleting', [TrainerPricingDiscountController::class, 'deleteDiscountRow']);

**A.2 : TrainerPricingDiscountController:**

**A.17.4** **getPriceCurrencyByUserId:**

**It is the function that responsible for getting Trainer currenct rows from online database.** Its route in the api.php Route::get('/get-trainer-price', [TrainerPricingCurrencyController::class, 'getPriceCurrencyByUserId']);

**A.2 : TrainerSubscriptionsController:**

**A.18.1** **traineeSubscribe:**

**It is the function that responsible for checking if the trainee is already subscribed to another trainer or same trainer before send him/her to the payment gateway.** Its route in the api.php Route::post('/trainer-subscribe', [TrainerSubscriptionsController::class, 'traineeSubscribe']);

**A.2 : TrainerSubscriptionsController:**

**A.18.2** **getTrainerTraineesByTok:**

**It is the function that responsible for getting active and inactive trainess of the trainer .** Its route in the api.php Route::get('/get-trainer-trainees', [TrainerSubscriptionsController::class, 'getTrainerTraineesByTok']);

**A.2 : TrainerSubscriptionsController:**

**A.18.3** **getTrainersDataForTrainee:**

**It is the function that responsible for getting trainers data for trainess.** Its route in the api.php Route::get('/get-trainers-Data-for-trainee', [TrainerSubscriptionsController::class, 'getTrainersDataForTrainee']);

**A.2 : TrainerSubscriptionsController:**

**A.18.4** **getTraineeSideData:**

**It is the function that responsible for getting trainee data by his id.** Its route in the api.php Route::get('/get-trainee-side-data', [TrainerSubscriptionsController::class, 'getTraineeSideData']);

**A.2 : TrainerSubscriptionsController:**

**A.18.5** **traineeSubscribeWithPayMob:**

**It is the function that responsible for checking that the trainee not subscribed to that trainer already or another trainer in this period** before send him/her to the paymob gateway(not used currently we use xpay).Its route in the api.php Route::post('/trainer-subscribe-with-paymob', [TrainerSubscriptionsController::class, 'traineeSubscribeWithPayMob']);

**A.2 : TrainerSubscriptionsController:**

**A.18.6** **traineeSubscribeWithXPay:**

**It is the function that responsible for checking that the trainee not subscribed to that trainer already or another trainer in this period before send him/her to xpay payment gateway** Its route in the api.php Route::post('/trainer-subscribe-with-xpay', [TrainerSubscriptionsController::class, 'traineeSubscribeWithXPay']);

**A.2 : TrainerSubscriptionsController:**

**A.18.7** **traineeSubscribeWithZeroPrice(free subscription function):**

**It is the function that responsible for checking that the trainee not subscribed to that trainer already or another trainer in this period then make free subscription for him/her** Its route in the api.php Route::post('/trainer-subscribe-free', [TrainerSubscriptionsController::class, 'traineeSubscribeWithZeroPrice']);

**A.2 : TrainerSubscriptionsController:**

**A.18.8** **trainerOnboardingCurrentTraineeForFree (free subscription function):**

**It is the function that responsible that trainer will use for onboarding his current trainee from outside the app into the app by entering trainee email.** Its route in the api.php Route::post('/trainer-Onboarding-Current-Trainee-For-Free', [TrainerSubscriptionsController::class, 'trainerOnboardingCurrentTraineeForFree']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.1** **checkTodayWorkouts:**

**It is the function that responsible for checking if the trainee made already workout today or also check if her made workout in other dates using in checking speKey** Its route in the api.php Route::get('/check-today-workouts', [TrainerTraineeBeginWorkoutsController::class, 'checkTodayWorkouts']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.2** **fetchLastDayStartWorkouts:**

**It is the function that responsible for getting the last workouts that trainee made in specific plan from online data.**Its route in the api.php Route::get('/fetch-last-day-start-workouts', [TrainerTraineeBeginWorkoutsController::class, 'fetchLastDayStartWorkouts']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.3** **fetchLastDayStartWorkoutsWithoutPlanId:**

**It is the function that responsible for getting the last workouts that trainee made not depended on specific plan but the last workouts from any plans from online data.**Its route in the api.php Route::get('/fetch-last-day-start-workouts-without-planId', [TrainerTraineeBeginWorkoutsController::class, 'fetchLastDayStartWorkoutsWithoutPlanId']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.4** **fetchSpecificWorkoutWithoutDeleting:**

**It is the function that responsible for getting specific plan but the last workouts from any plans from online data.**Its route in the api.php Route::get('/fetch-specific-workout-without-deleting', [TrainerTraineeBeginWorkoutsController::class, 'fetchSpecificWorkoutWithoutDeleting']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.4** **insertOrUpdateTrainerTraineeBeginWorkouts:**

**It is the function that responsible for inserting workouts that trainee did into online data.**Its route in the api.php Route::post('/trainer-trainee-begin-workout-insert', [TrainerTraineeBeginWorkoutsController::class, 'insertOrUpdateTrainerTraineeBeginWorkouts']);

**A.2 : TrainerTraineeBeginWorkoutsController:**

**A.19.4** **fetchAllPerformedWorkouts:**

**It is the function that responsible for getting all performed workouts by trainee into online data.**Its route in the api.php Route::get('/fetch-all-performed-workouts', [TrainerTraineeBeginWorkoutsController::class, 'fetchAllPerformedWorkouts']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.1** **insertTrainerTraineeMealsPlanDayMeals:**

**It is the function that responsible for inserting meals that trainer assigned to trainee into online data.**Its route in the api.php Route::post('/trainer-trainee-meal-plan-day-insert', [TrainerTraineeMealsPlanDaysController::class, 'insertTrainerTraineeMealsPlanDayMeals']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.2** **insertTrainerTraineeMealsPlanDayManyDays:**

**It is the function that responsible for inserting meals for many days that trainer assigned to trainee into online data.**Its route in the api.php Route::post('/trainer-trainee-meal-plan-day-insert', [TrainerTraineeMealsPlanDaysController::class, 'insertTrainerTraineeMealsPlanDayMeals']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.3** **insertOrUpdatetrainerTraineeCheckBoxOFDayMeal:**

**It is the function that responsible for make check signs by trainee on meals that he eat into online data.**Its route in the api.php Route::post('/trainer-trainee-plan-days-insert-Or-Update-Check-Box-OF-Day-Meal', [TrainerTraineeMealsPlanDaysController::class, 'insertOrUpdatetrainerTraineeCheckBoxOFDayMeal']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.4** **editTrainerTraineeMealsPlanDayManyDays:**

**It is the function that responsible for editing meals that trainer assign for trainee for many days.**Its route in the api.php Route::post('/trainer-trainee-meals-plan-day-edit-many-days', [TrainerTraineeMealsPlanDaysController::class, 'editTrainerTraineeMealsPlanDayManyDays']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.5** **makeTheMealOfDayCheck:**

**It is the function that responsible for make check sign on one meal in day that trainee eat by himself .**Its route in the api.php Route::post('/make-The-Meal-Of-Day-Check', [TrainerTraineeMealsPlanDaysController::class, 'makeTheMealOfDayCheck']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.6** **getTrainerTraineeMealPlanDays:**

**It is the function that responsible for getting trainer trainee meals days in the plan that assigned by trainer from online data .**Its route in the api.php Route::get('/get-trainer-trainee-meal-plan-days-for-day-new-style', [TrainerTraineeMealsPlanDaysController::class, 'getTrainerTraineeMealPlanDaysForDayNewStyle']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.7** **deleteTrainerTraineeMealPlanDayRow:**

**It is the function that responsible for deleting trainee meals day assaigned by trainer.**Its route in the api.php Route::post('/trainer-trainee-meal-plan-day-deleting', [TrainerTraineeMealsPlanDaysController::class, 'deleteTrainerTraineeMealPlanDayRow']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.8** **insertTrainerTraineeMealPlanDaysNewStylePlus (new style):**

**It is the function that responsible for insert meals by trainer to the trainee by press on plus sign beside every meal.**Its route in the api.php Route::post('/trainer-trainee-meal-plan-day-deleting', [TrainerTraineeMealsPlanDaysController::class, 'deleteTrainerTraineeMealPlanDayRow']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.9** **removeOneMealFromMealPlanDaysNewStyleMinus (new style):**

**It is the function that responsible for remove on meal that assigned by trainer to the trainee by press on minus sign beside every meal.**Its route in the api.php Route::post('/trainer-trainee-meal-plan-day-deleting', [TrainerTraineeMealsPlanDaysController::class, 'deleteTrainerTraineeMealPlanDayRow']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.10** **getTrainerTraineeMealPlanDaysForDayNewStyle:**

**It is the function that responsible for getting on day meals that assigned by trainer to the trainee .**

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.20.11** **restartAllMealsByChangingOneToZero:**

**It is the function that responsible for make all the meals that checked by trainee uncheckd.**

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.1** **insertTrainerTraineeMealsPlans:**

**It is the function that responsible for inserting meals plans assigned by trainer to trainee into online database.**Its route in the api.php Route::post('/trainer-plans-meals-insert', [TrainerTraineeMealsPlansController::class, 'insertTrainerTraineeMealsPlans']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.2** **updateTrainerTraineeMealsPlans:**

**It is the function that responsible for updating meals plans assigned by trainer to trainee into online database.**Its route in the api.php Route::post('/trainer-plans-meals-update', [TrainerTraineeMealsPlansController::class, 'updateTrainerTraineeMealsPlans']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.3** **insertUnlimitedTrainerTraineeMealsPlans:**

**It is the function that responsible for inserting unlimited meals plans assigned by trainer to trainee into online database.**Its route in the api.php Route::post('/trainer-plans-meals-insert-unlimited', [TrainerTraineeMealsPlansController::class, 'insertUnlimitedTrainerTraineeMealsPlans']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.4** **updateUnlimitedTrainerTraineeMealsPlans:**

**It is the function that responsible for updating unlimited meals plans assigned by trainer to trainee into online database.**Its route in the api.php Route::post('/trainer-plans-meals-insert-unlimited', [TrainerTraineeMealsPlansController::class, 'insertUnlimitedTrainerTraineeMealsPlans']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.5** **getTrainerTraineeMealsPlans:**

**It is the function that responsible for getting the meals plans assigned by trainer to trainee from online database.**Its route in the api.php Route::get('/get-trainer-trainee-meals-plans', [TrainerTraineeMealsPlansController::class, 'getTrainerTraineeMealsPlans']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.6** **deleteTrainerTraineeMealRow:**

**It is the function that responsible for deleting the meals plans assigned by trainer to trainee from online database.**Its route in the api.php Route::post('/trainer-trainee-meals-plan-deleting', [TrainerTraineeMealsPlansController::class, 'deleteTrainerTraineeMealRow']);

**A.2 : TrainerTraineeMealsPlanDaysController:**

**A.21.7** **getMealPlanDaysForCurrentDate:**

**It is the function that responsible for getting the meals plans assigned by trainer to trainee for current date from online database.**Its route in the api.php Route::get('/get-trainer-trainee-meal-current-plan-days', [TrainerTraineeMealsPlanDaysController::class, 'getMealPlanDaysForCurrentDate']);

**A.2 : TrainerTraineeQuestionnairePageController:**

**A.22.1** **insertTrainerTraineeQuestionnairePageRow:**

**It is the function that responsible for inserting questionnaire answers that made by trainees on the trainer questions the meals plans assigned by trainer to trainee for current date into online database.**Its route in the api. Route::post('/trainer-trainee-Questionnaire-answers-insert', [TrainerTraineeQuestionnairePageController::class, 'insertTrainerTraineeQuestionnairePageRow']);

**A.2 : TrainerTraineeQuestionnairePageController:**

**A.22.2** **getLastQuestionnaireForTrainee:**

**It is the function that responsible for getting last questionnaire answers that made by trainees on the trainer questions the meals plans assigned by trainer to trainee for current date from online database.**Its route in the api. Route::get('/get-Last-Questionnaire-For-Trainee', [TrainerTraineeQuestionnairePageController::class, 'getLastQuestionnaireForTrainee']);

**A.2 : TrainerTraineeQuestionnairePageController:**

**A.23.1** **insertOrUpdatetrainerTrainee****RequiredMacros:**

**It is the function that responsible for inserting Required Macros by trainer to the trainees into online database.**Its route in the api. Route::post('/trainer-trainee-add-or-update-required-macros', [trainerTraineeRequiredMacrosController::class, 'insertOrUpdatetrainerTraineeRequiredMacros']);

**A.2 : TrainerTraineeQuestionnairePageController:**

**A.23.2** **fetchLastTrainerTraineeRequiredMacros:**

**It is the function that responsible for getting Required Macros by trainer to the trainees from online database.**Its route in the api. Route::post('/trainer-trainee-add-or-update-required-macros', [trainerTraineeRequiredMacrosController::class, 'insertOrUpdatetrainerTraineeRequiredMacros']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.1** **insertTrainerTraineeTodayMeals:**

**It is the function that responsible for inserting today meals into online database.**Its route in the api. Route::post('/trainer-trainee-today-meal-insert', [TrainerTraineeTodayMealsController::class, 'insertTrainerTraineeTodayMeals']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.2** **insertTrainerTraineeTodayMealsNewStylePlus (new style):**

**It is the function that responsible for inserting today meals into online database.**Its route in the api. Route::post('/trainer-trainee-today-meal-insert-new-style-plus', [TrainerTraineeTodayMealsController::class, 'insertTrainerTraineeTodayMealsNewStylePlus']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.3** **removeOneMealFromTodayMealsNewStyleMinus:**

**It is the function that responsible for deleting one meal from today meals from online database.**Its route in the api. Route::post('/trainer-trainee-today-meal-insert-new-style-plus', [TrainerTraineeTodayMealsController::class, 'insertTrainerTraineeTodayMealsNewStylePlus']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.4** **insertTrainerTraineeTodayMealsNewStylePlus****FromTrainerAddMealTodayPag:**

**It is the function that responsible for inserting today meals by pressing plus button from From Trainer Add Meal into online database.**Its route in the api. Route::post('/add-One-Meal-From-TodayMeals-new-style-plus-from-trainer-add-today-page', [TrainerTraineeTodayMealsController::class, 'insertTrainerTraineeTodayMealsNewStylePlusFromTrainerAddMealTodayPag']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.5** **removeOneMealFromTodayMealsNewStyleMinuFromTrainerTodayPage:**

**It is the function that responsible for deleting one meal from today meals page.**Its route in the api. Route::post('/remove-One-Meal-From-TodayMeals-new-style-minus-from-trainer-today-page', [TrainerTraineeTodayMealsController::class, 'removeOneMealFromTodayMealsNewStyleMinuFromTrainerTodayPage']);  
  
**A.2 : TrainerTraineeTodayMealsController:**

**A.24.6** **removeOneMealFromTodayMealsNewStyleMinus****WithDateByUncheck:**

**It is the function that responsible for deleting one meal from today meals page With Date By Uncheck.**Its route in the api. Route::post('/remove-One-Meal-From-TodayMeals-new-style-minus', [TrainerTraineeTodayMealsController::class, 'removeOneMealFromTodayMealsNewStyleMinusWithDateByUncheck']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.7** **getTrainerTraineeTodayMeals:**

**It is the function that responsible for getting meals from online database.**Its route in the api. Route::get('/get-trainer-trainee-today-meals', [TrainerTraineeTodayMealsController::class, 'getTrainerTraineeTodayMeals']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.8** **getTrainerTraineeTodayMealsForTodayMealPage****InTrainerManageMeals:**

**It is the function that responsible for getting today meals InTrainer Manage Meals from online database.**Its route in the api. Route::get('/get-trainer-trainee-today-meals-for-today-meal-page-in-trainer-manage-meals', [TrainerTraineeTodayMealsController::class, 'getTrainerTraineeTodayMealsForTodayMealPageInTrainerManageMeals']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.9** **fetchAllPerformedMeals:**

**It is the function that responsible for getting all performed today from online database.**Its route in the api. Route::get('/fetch-all-performed-meals', [TrainerTraineeTodayMealsController::class, 'fetchAllPerformedMeals']);

**A.2 : TrainerTraineeTodayMealsController:**

**A.24.10** **getOneMealFromTrainerTraineeTodayMeals:**

**It is the function that responsible for getting one meal from today meals from online database.**Its route in the api. Route::get('/get-One-Meal-From-Trainer-Trainee-Today-Meals', [TrainerTraineeTodayMealsController::class, 'getOneMealFromTrainerTraineeTodayMeals']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.1** **insertTrainerTrainee****WorkoutsPlanDayWorkouts:**

**It is the function that responsible for inserting Workouts Plan session into online database.**Its route in the api Route::post('/trainer-trainee-plan-day-insert', [TrainerTraineeWorkoutsPlanDaysController::class, 'insertTrainerTraineeWorkoutsPlanDayWorkouts']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.2** **insertTrainerTraineeWorkoutsPlanDayWorkoutsManyDays:**

**It is the function that responsible for inserting Workouts Plan sessions into online database.**Its route in the Route::post('/trainer-trainee-plan-day-insert-many-days', [TrainerTraineeWorkoutsPlanDaysController::class, 'insertTrainerTraineeWorkoutsPlanDayWorkoutsManyDays']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.3** **editTrainerTraineeWorkoutsPlanDayWorkoutsManyDays:**

**It is the function that responsible for editing Workouts Plan sessions in online database.**Its route in the Route::post('/trainer-trainee-plan-day-edit-many-days', [TrainerTraineeWorkoutsPlanDaysController::class, 'editTrainerTraineeWorkoutsPlanDayWorkoutsManyDays']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.4** **updateTrainerTraineeWorkoutsPlans:**

**It is the function that responsible for updating Workouts Plans assigned by trainer to the trainee in online database.**Its route in the Route::post('/trainer-plans-update', [TrainerTraineeWorkoutsPlansController::class, 'updateTrainerTraineeWorkoutsPlans']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.5** **getTrainerTraineePlanDays:**

**It is the function that responsible for getting Workouts Plans sessions assigned by trainer to the trainee from online database.**Its route in the Route::get('/get-trainer-trainee-plan-days', [TrainerTraineeWorkoutsPlanDaysController::class, 'getTrainerTraineePlanDays']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.6** **getTrainerTraineePlanDaysWithoutPlanId:**

**It is the function that responsible for getting Workouts Plans sessions assigned by trainer to the trainee from any plan from online database.**Its route in the Route::get('/get-trainer-trainee-plan-days-without-planId', [TrainerTraineeWorkoutsPlanDaysController::class, 'getTrainerTraineePlanDaysWithoutPlanId']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.7** **deleteTrainerTraineePlanDayRow:**

**It is the function that responsible for deleting Workouts Plan session row assigned by trainer to the trainee from online database.**Its route in the Route::post('/trainer-trainee-plan-day-deleting', [TrainerTraineeWorkoutsPlanDaysController::class, 'deleteTrainerTraineePlanDayRow']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.8** **skipMarkWorkoutOrDone:**

**It is the function that responsible for making Workouts Plan session row assigned by trainer to the trainee skipped or done into online database.**Its route in the Route::post('/trainer-trainee-plan-day-skip-workout-or-done', [TrainerTraineeWorkoutsPlanDaysController::class, 'skipMarkWorkoutOrDone']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.9** **restartWorkoutByChangingToActive:**

**It is the function that responsible for restart Workouts Plan session row assigned by trainer to the trainee that has been skipped or done into online database.**Its route in the Route::post('/trainer-trainee-plan-day-restart-workout-to-active', [TrainerTraineeWorkoutsPlanDaysController::class, 'restartWorkoutByChangingToActive']);

**A.2 : TrainerTraineeWorkoutsPlanDaysController:**

**A.25.10** **restartAllWorkoutByChangingToActive:**

**It is the function that responsible for restart Workouts Plan sessions assigned by trainer to the trainee that has been skipped or done into online database.**Its route in the Route::post('/trainer-trainee-plan-day-restart-all-workouts-to-active', [TrainerTraineeWorkoutsPlanDaysController::class, 'restartAllWorkoutByChangingToActive']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.1** **insertTrainerTraineeWorkoutsPlans:**

**It is the function that responsible for inserting Workouts Plans assigned by trainer to the trainee into online database.**Its route in the Route::post('/trainer-plans-insert', [TrainerTraineeWorkoutsPlansController::class, 'insertTrainerTraineeWorkoutsPlans']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.2** **insertUnlimitedPlansPublicWorkoutsPlans:**

**It is the function that responsible for inserting unlimited Workouts Plans assigned by trainer to the trainee into online database.**Its route in the Route::post('/trainer-plans-insert-unlimited', [TrainerTraineeWorkoutsPlansController::class, 'insertUnlimitedPlansPublicWorkoutsPlans']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.3** **updateUnlimitedPlansPublicWorkoutsPlans:**

**It is the function that responsible for updating unlimited Workouts Plans assigned by trainer to the trainee into online database.**Its route in the Route::post('/trainer-plans-update-unlimited', [TrainerTraineeWorkoutsPlansController::class, 'updateUnlimitedPlansPublicWorkoutsPlans']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.4** **updateTrainerTraineeWorkoutsPlans:**

**It is the function that responsible for updating Workouts Plans assigned by trainer to the trainee into online database.**Its route in the Route::post('/trainer-plans-update', [TrainerTraineeWorkoutsPlansController::class, 'updateTrainerTraineeWorkoutsPlans']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.5** **getTrainerTraineePlans:**

**It is the function that responsible for getting Workouts Plans assigned by trainer to the trainee from online database.**Its route in the Route::get('/get-trainer-trainee-plans', [TrainerTraineeWorkoutsPlansController::class, 'getTrainerTraineePlans']);

**A.2 : TrainerTraineeWorkoutsPlansController:**

**A.26.6** **deleteTrainerTraineeRow:**

**It is the function that responsible for deleting Workouts Plan assigned by trainer to the trainee from online database.**Its route in the Route::post('/trainer-trainee-plan-deleting', [TrainerTraineeWorkoutsPlansController::class, 'deleteTrainerTraineeRow']);

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.1** **insertTrainerWorkoutsPredefinedPlanDays:**

**It is the function that responsible for inserting predefined Workouts Plan sessions assigned by trainer to the trainee into online database.**Its route in the Route::post('/trainer-workouts-predefined-plan-day-insert', [TrainerWorkoutsPredefinedPlanDaysController::class, 'insertTrainerWorkoutsPredefinedPlanDays']);

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.2** **getTrainerWorkoutsPredefinedPlanDays:**

**It is the function that responsible for getting predefined Workouts Plan sessions assigned by trainer to the trainee from online database.**Its route in the Route::get('/get-trainer-workouts-predefined-plan-days', [TrainerWorkoutsPredefinedPlanDaysController::class, 'getTrainerWorkoutsPredefinedPlanDays']);

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.3** **getTrainerPredefinedPlansWithItsDays:**

**It is the function that responsible for getting predefined Workouts Plans and its sessions assigned by trainer to the trainee from online database.**Its route in the Route::get('/get-trainer-workouts-predefined-plans-with-its-days', [TrainerWorkoutsPredefinedPlanDaysController::class, 'getTrainerPredefinedPlansWithItsDays']);

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.4** **deleteTrainerWorkoutsPredefinedPlanDaysRow:**

**It is the function that responsible for deleting predefined Workouts Plan sessions assigned by trainer to the trainee from online database.**Its route in the Route::post('/trainer-workouts-predefined-plan-day-deleting', [TrainerWorkoutsPredefinedPlanDaysController::class, 'deleteTrainerWorkoutsPredefinedPlanDaysRow']);

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.5** **skipMarkWorkoutOrDone:**

**It is the function that responsible for skipping or done predefined Workouts Plan sessions assigned by trainer to the trainee into online database.**

**A.2 : TrainerWorkoutsPredefinedPlanDaysController:**

**A.27.6** **restartWorkoutByChangingToActive:**

**It is the function that responsible for restart predefined Workouts Plan sessions assigned by trainer to the trainee that has been done or skipped from online database.**

**A.2 : workoutsController:**

**A.27.7** **insertWorkouts:**

**It is the function that responsible for insert all workouts from local database to online database. Its route in api.php file is Route::post('/workouts-insert-data', [workoutsController::class, 'insertWorkouts']);**

**A.2 : workoutsController:**

**A.27.8** **insertWorkoutsWithoutMedia:**

**It is the function that responsible for insert all workouts without media from local database to online database. Its route in api.php file is Route::post('/workouts-insert-data-without-media', [workoutsController::class, 'insertWorkoutsWithoutMedia']);**

**A.2 : xpayController:**

**A.28.1** **payWithXPay:**

**It is the function that responsible for paying trainess their subscriptions to trainers by xpay payement gateway this part one from the process of paying . Its route in api.php file is Route::post('/payWithXPay', [xpayController::class, 'payWithXPay'])->name('payWithXPay');**

**A.2 : xpayController:**

**A.28.2** **xpayCallback:**

**It is the function that responsible for paying trainess their subscriptions to trainers by xpay payement gateway this part two from the process of paying where it entring the data into two tables TrainerTraineePayments and TrainerSubscriptions. Its route in api.php file is Route::post('/xpayCallbackUrl', [xpayController::class, 'xpayCallback']);**

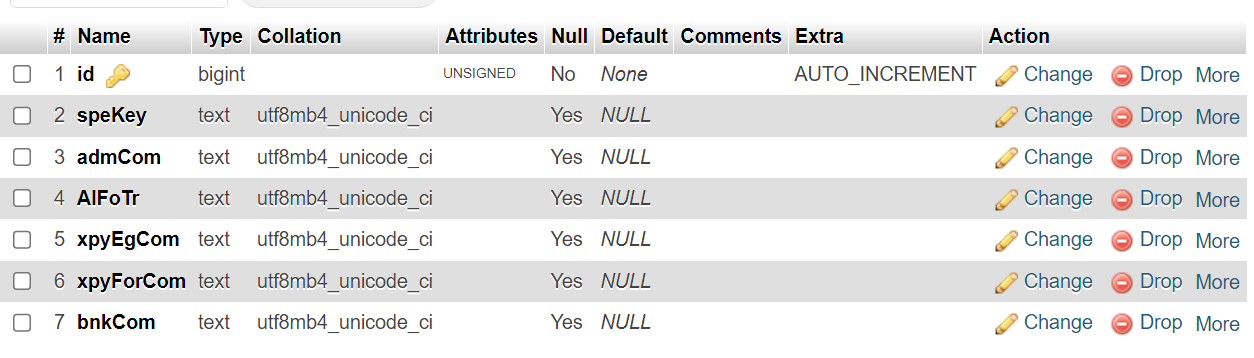
**A.2 : xpayController:**

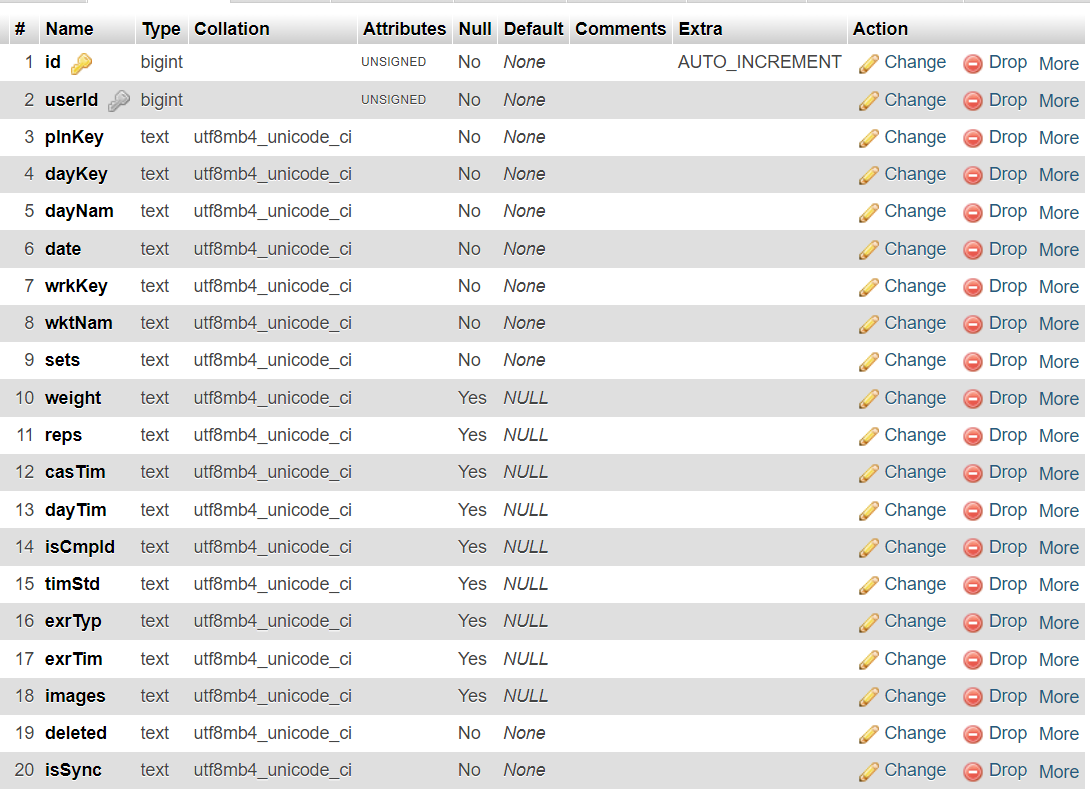
**A.28.3** **xpayToTrainerPayHisMoney:**

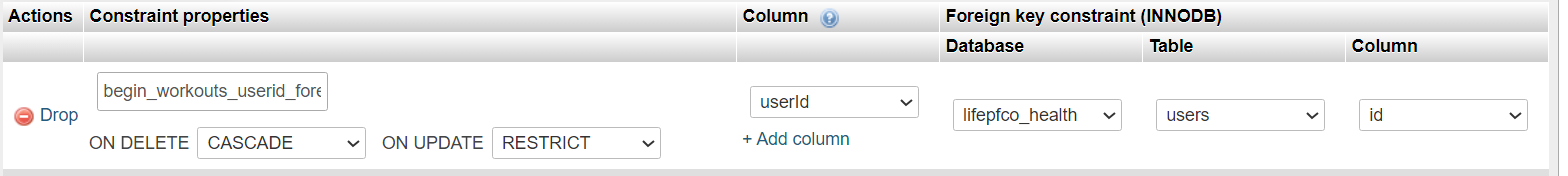
**It is the function that responsible for record in the database the payout that made to the trainer by transferring his money by the bank. Its route in api.php file is Route::post('/xpay-to-trainer-pay-his-money', [xpayController::class, 'xpayToTrainerPayHisMoney']);**

1. **Database Schema Diagrams:**

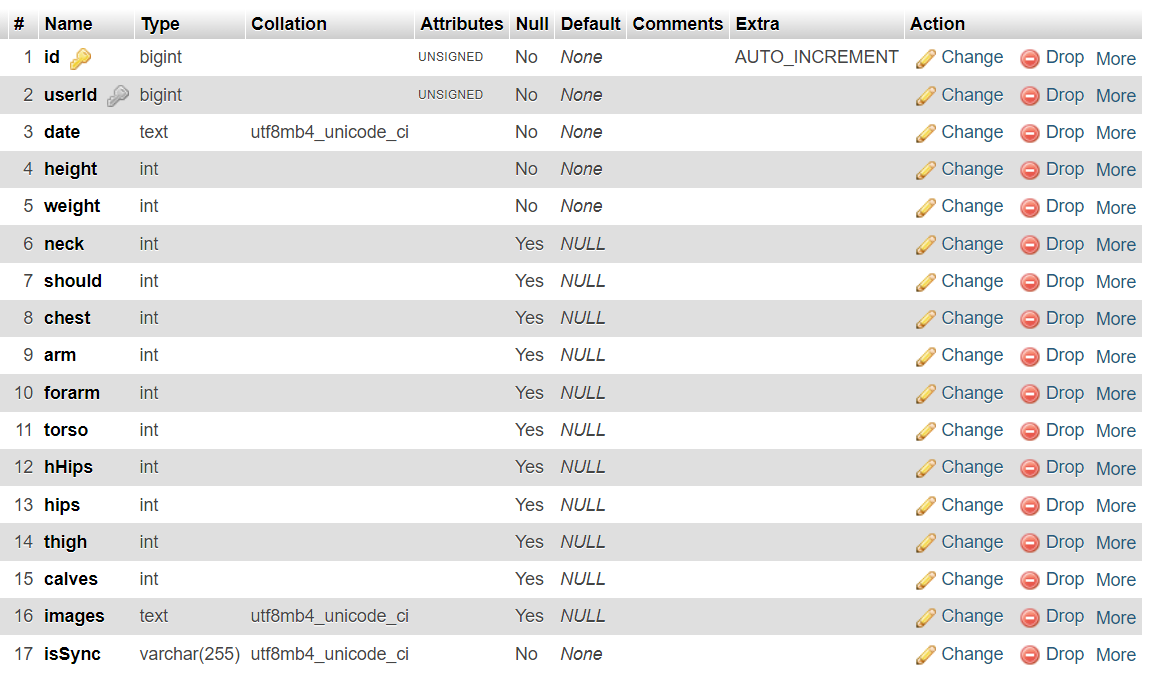
**2-1:** **admin\_settings\_app table:**

****

**2-2:** **begin\_workouts table:**

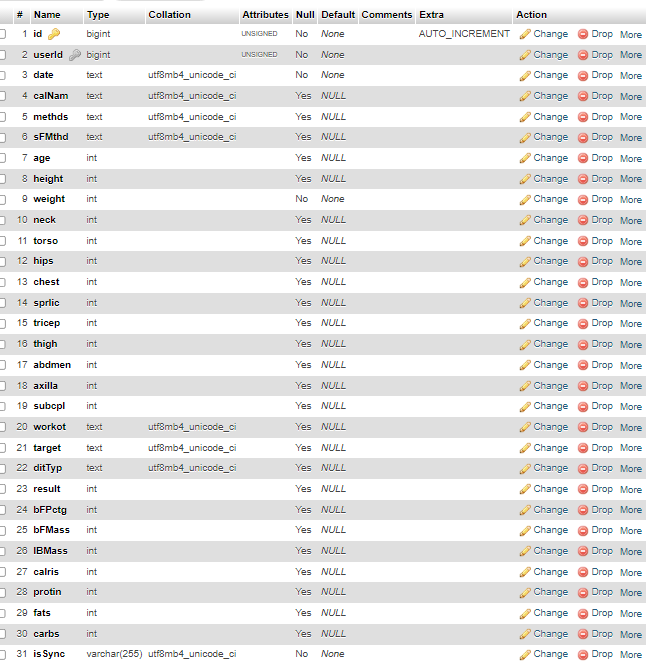
****

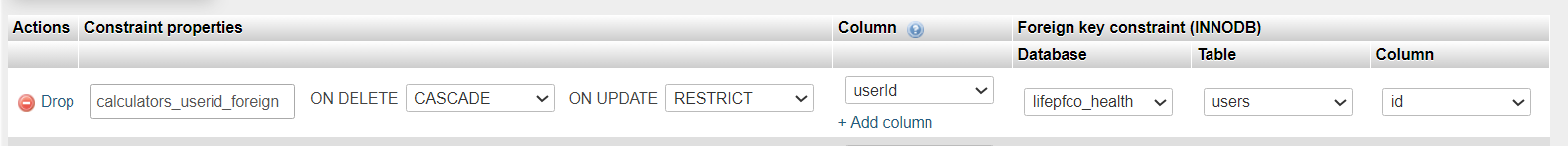
**2-3:** **body\_stats\_and\_measurements table:**

****

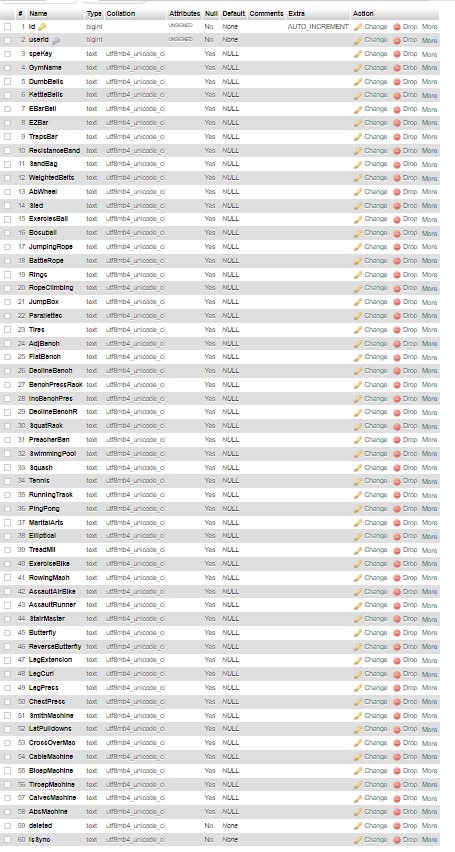
****

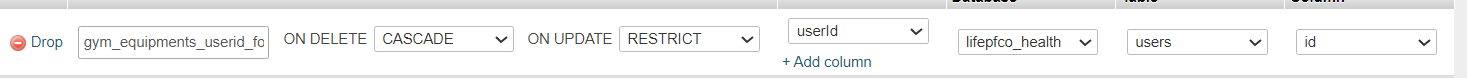
**2-4:** **calculators table:**

****

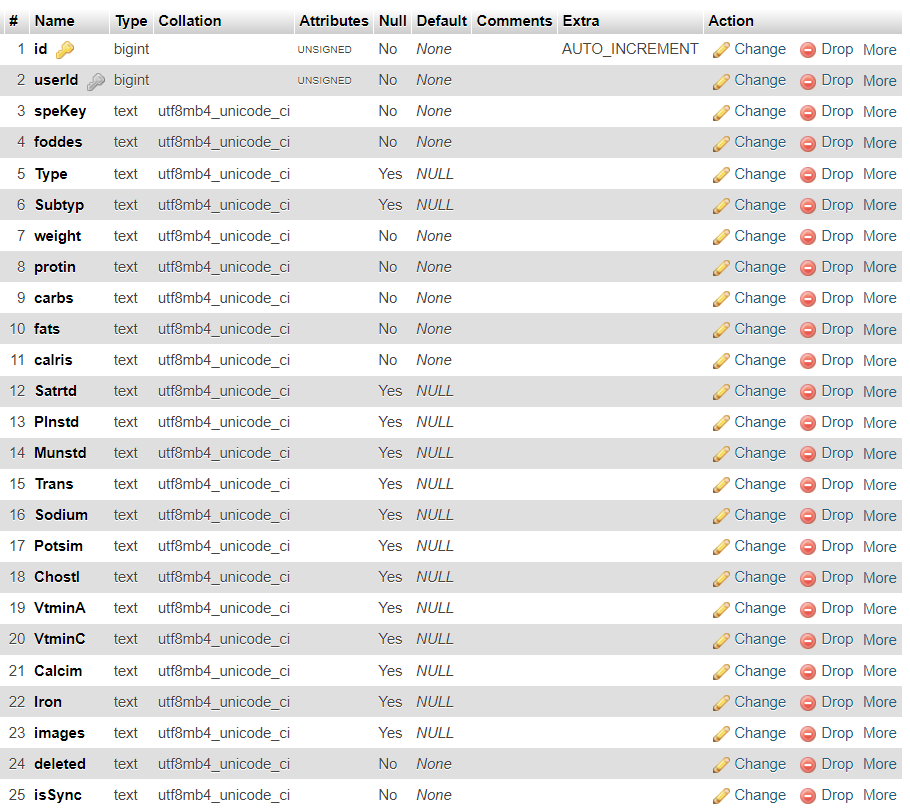
****

**2-5:** **gym\_equipments table:**

****

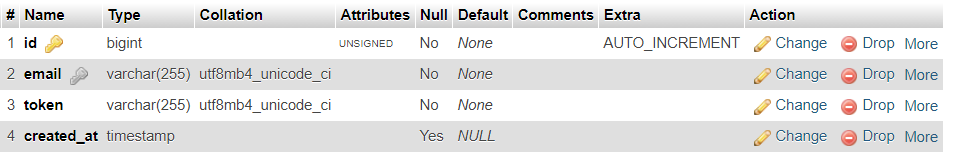
****

**2-6:** **list\_of\_foods table:**

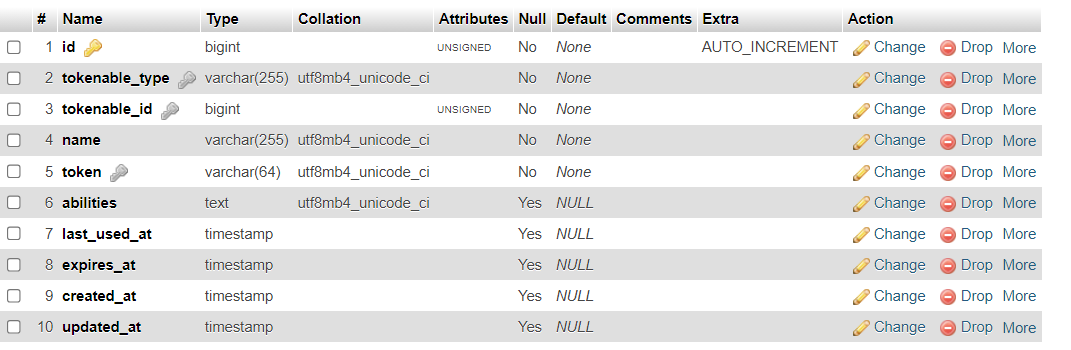
****

****

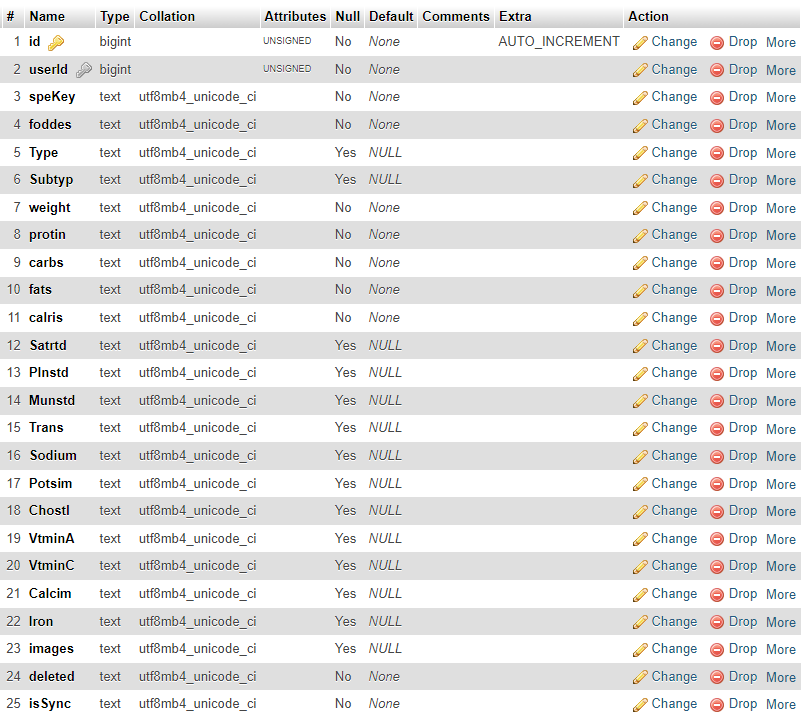
**2-7:** **password\_reset\_tokens table:**

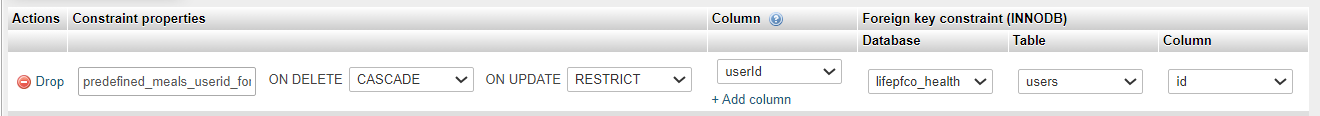
****

**2-8:** **personal\_access\_tokens table:**

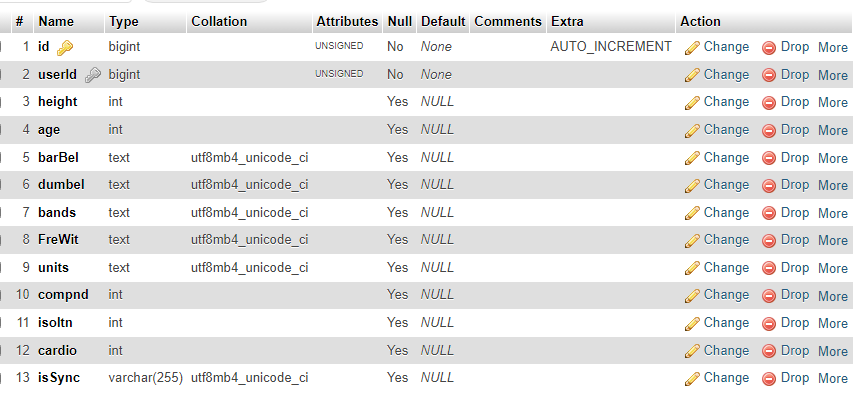
****

**2-9: predefined\_meals table:**

****

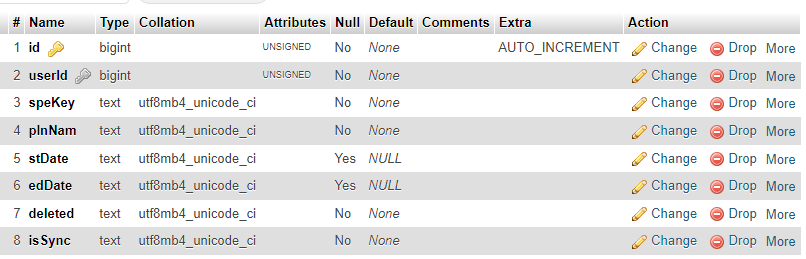
****

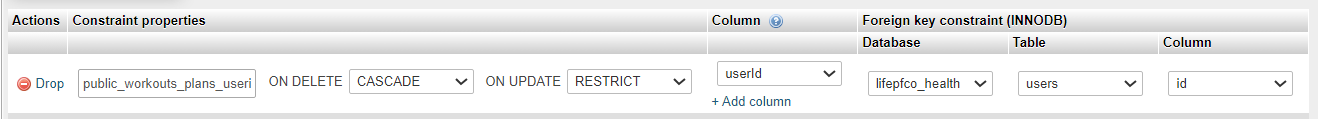
**2-10: public\_settings table:**

****

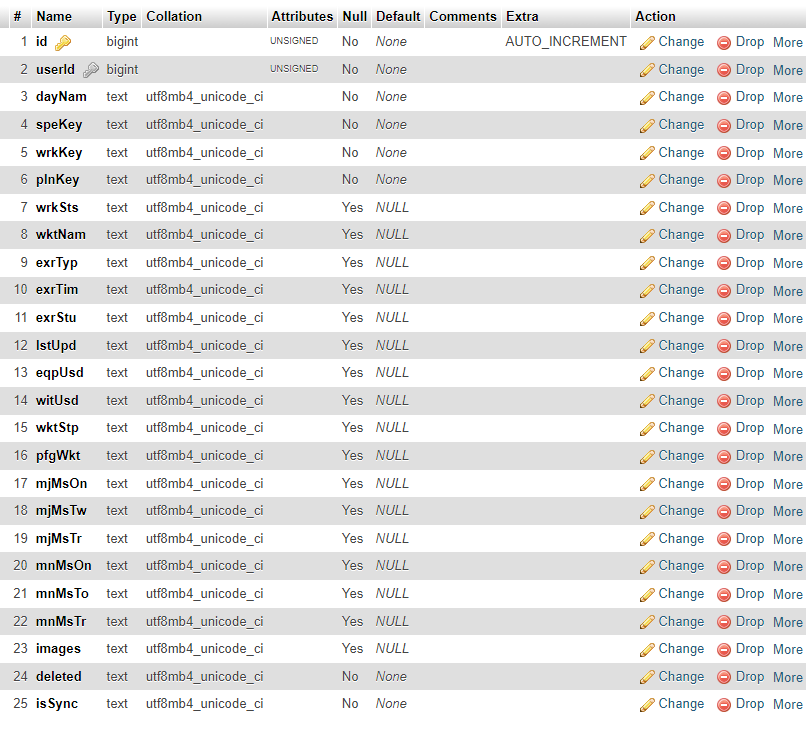
****

**2-11: public\_workouts\_plans table:**

****

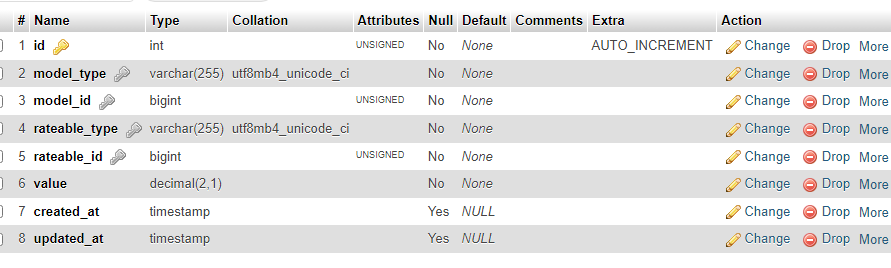
****

**2-12: public\_workouts\_plan\_days table:**

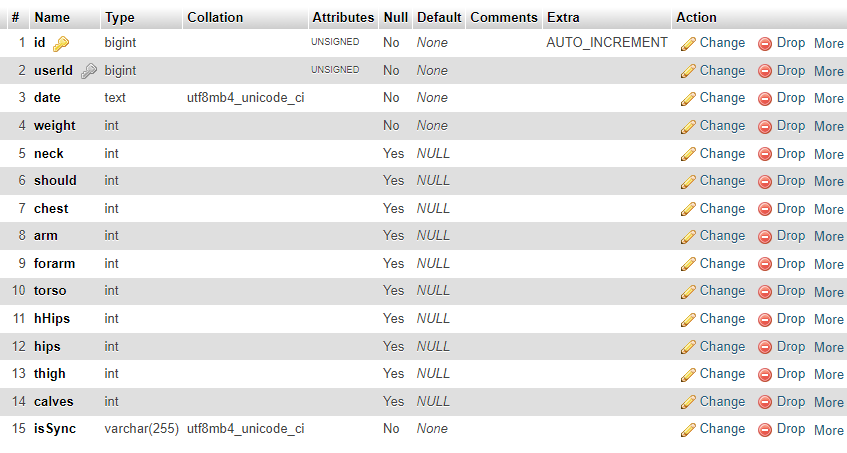
****

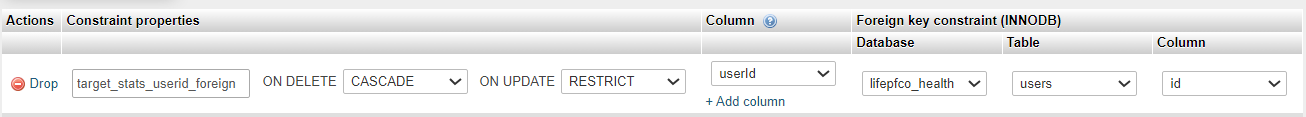
****

**2-13: ratings table:**

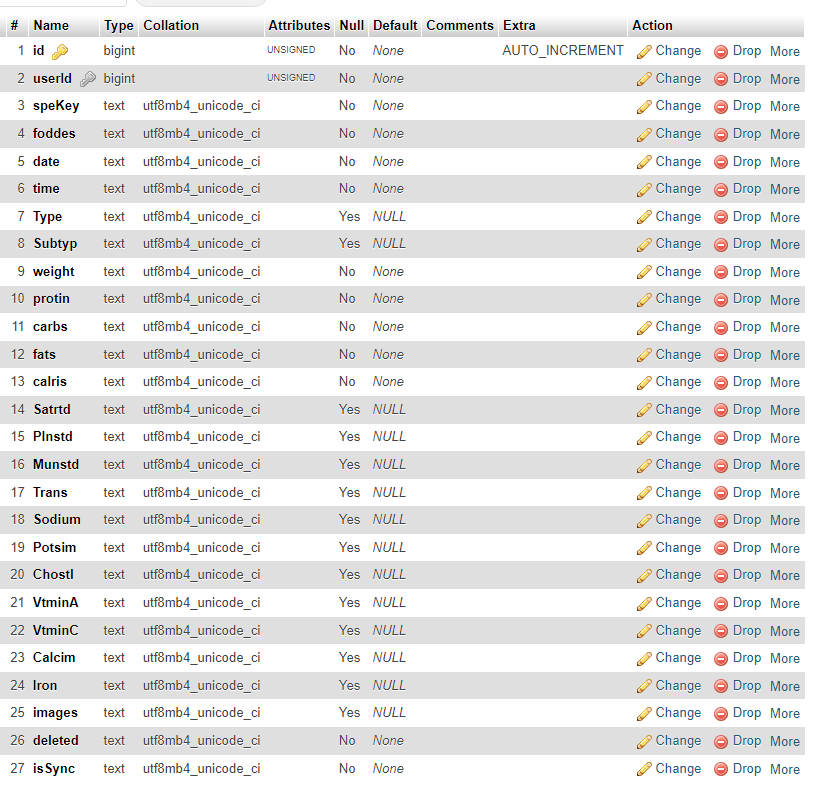
****

**2-14: target\_stats table:**

****

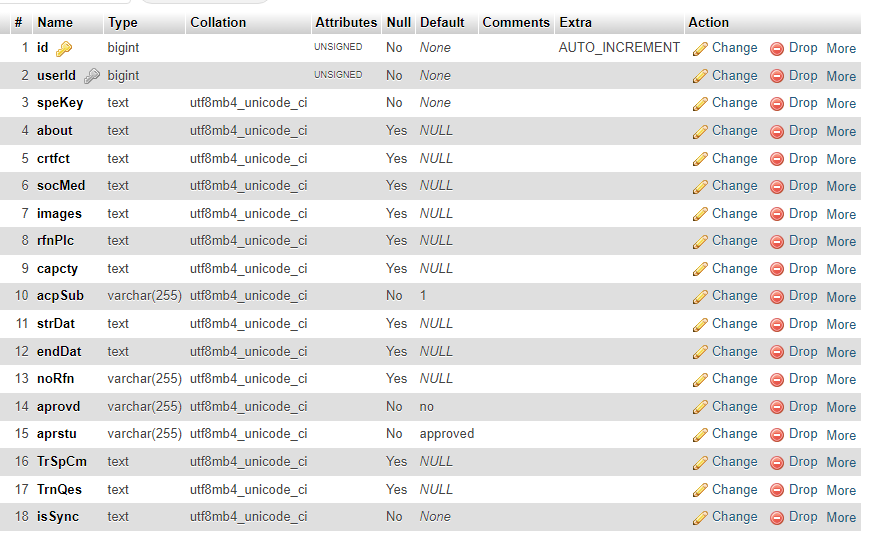
****

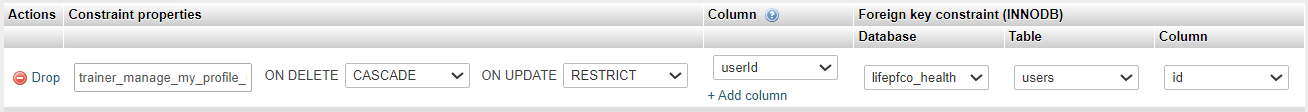
**2-15: today\_meals table:**

****

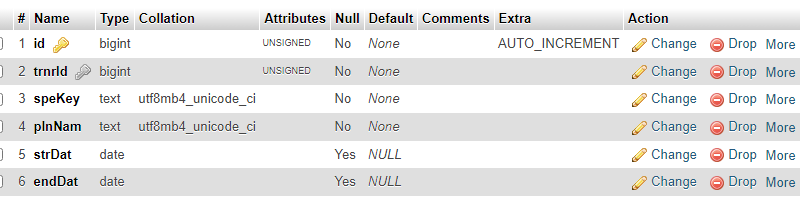
****

**2-16 : trainer\_manage\_my\_profile table:**

****

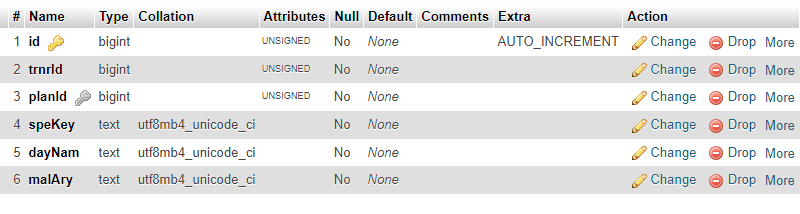
****

**2-17 : trainer\_meals\_predefined\_plans table:**

****

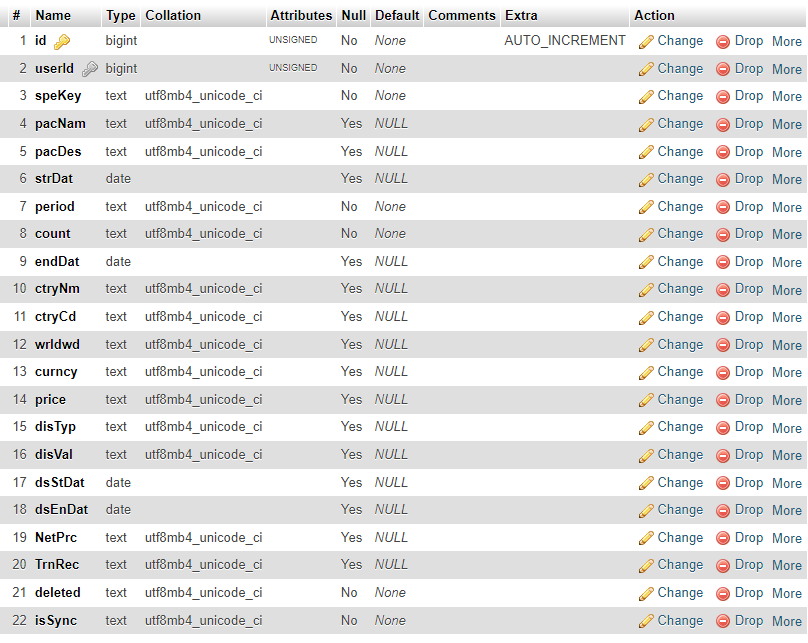
****

**2-18 : trainer\_meals\_predefined\_plan\_days table:**

****

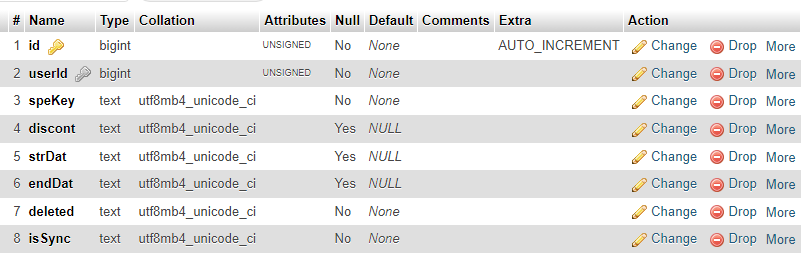
****

**2-19 : trainer\_pricing\_currency table:**

****

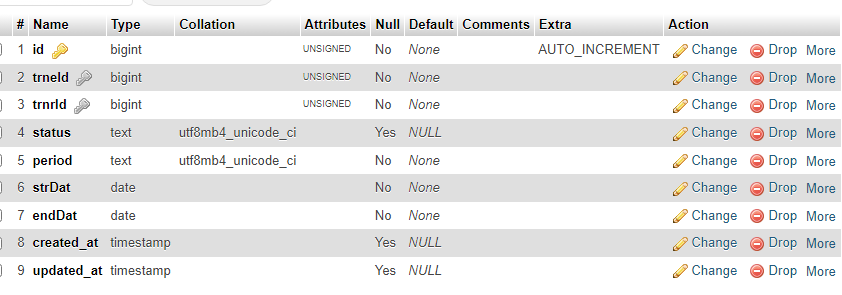
****

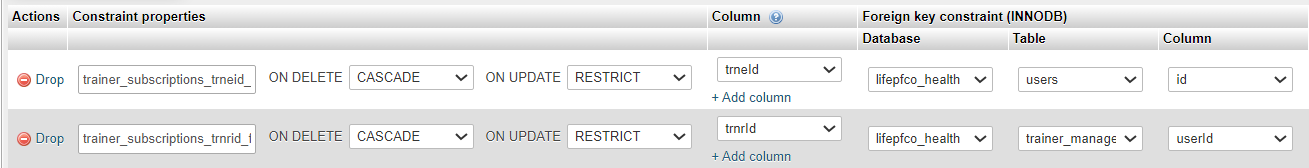
**2-20: trainer\_pricing\_discount table:**

****

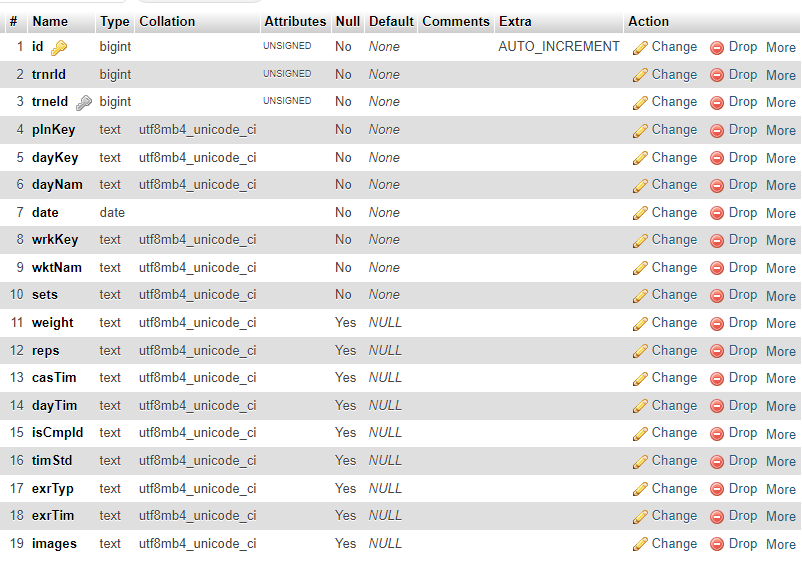
****

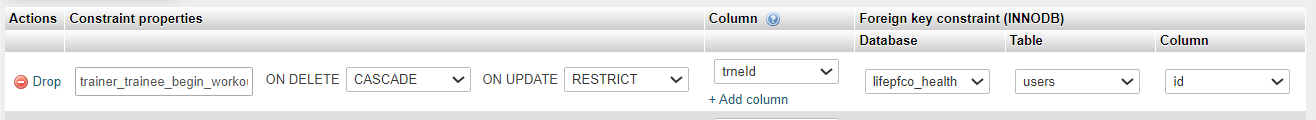
**2-21 : trainer\_subscriptions table:**

****

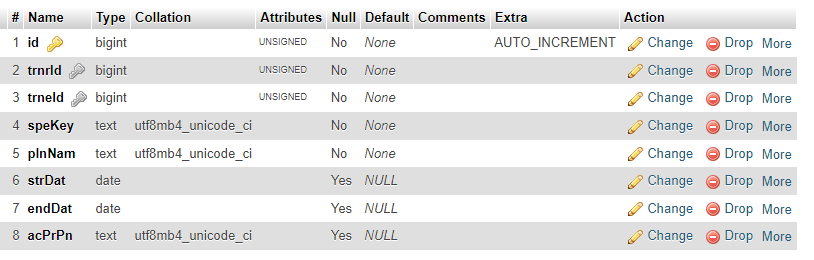
****

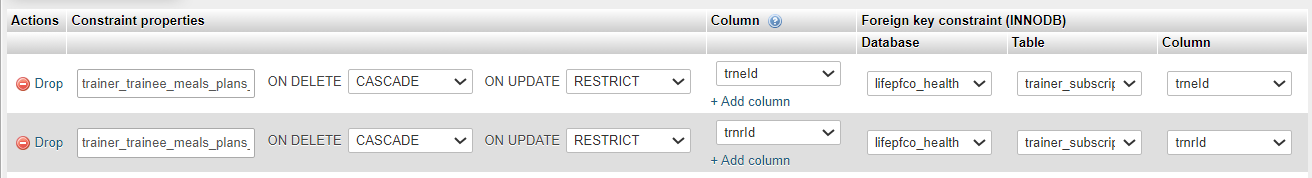
**2-22 : trainer\_trainee\_begin\_workouts table:**

****

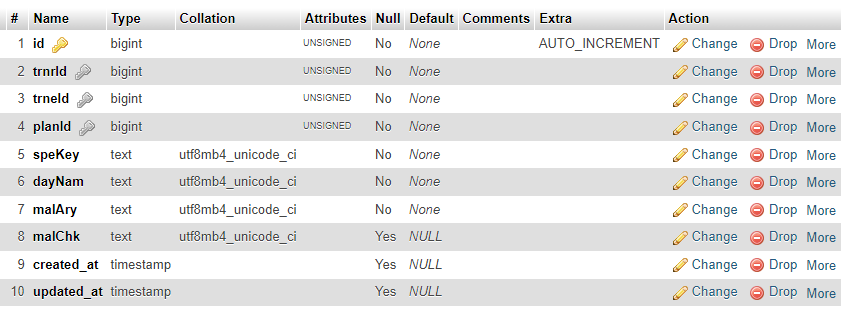
****

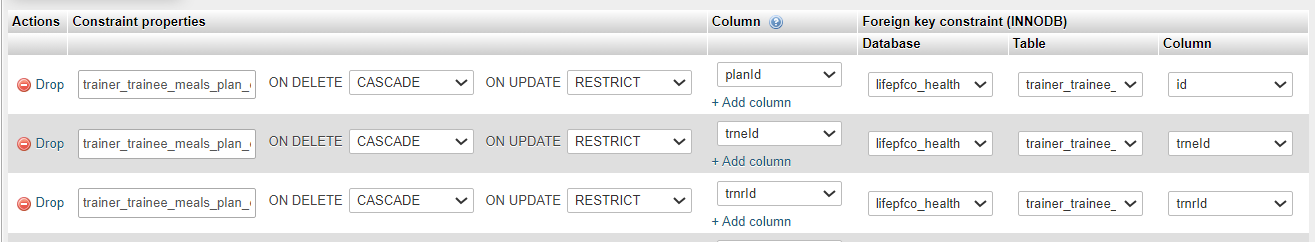
**2-23 : trainer\_trainee\_meals\_plans table:**

****

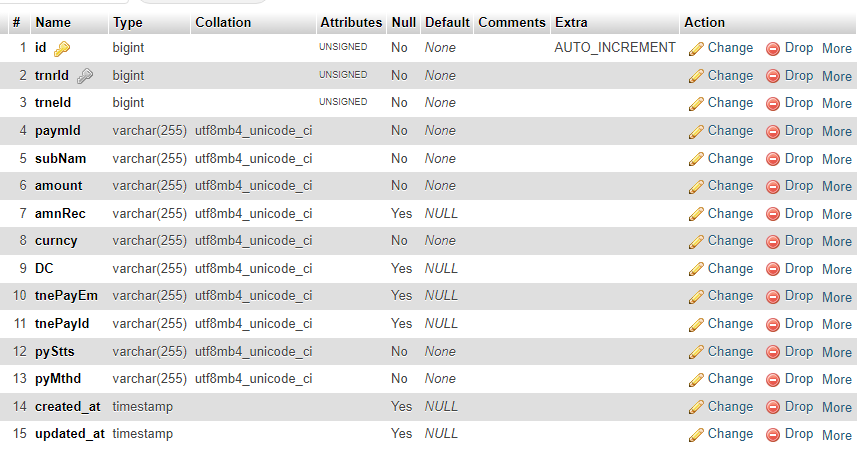
****

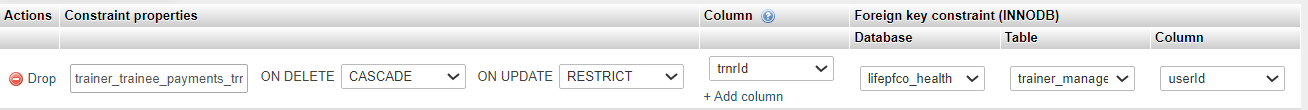
**2-24 : trainer\_trainee\_meals\_plan\_days table:**

****

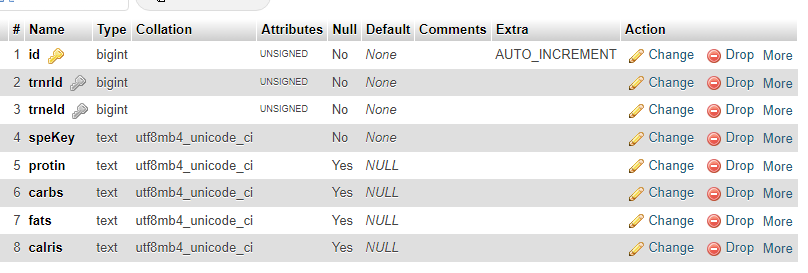
****

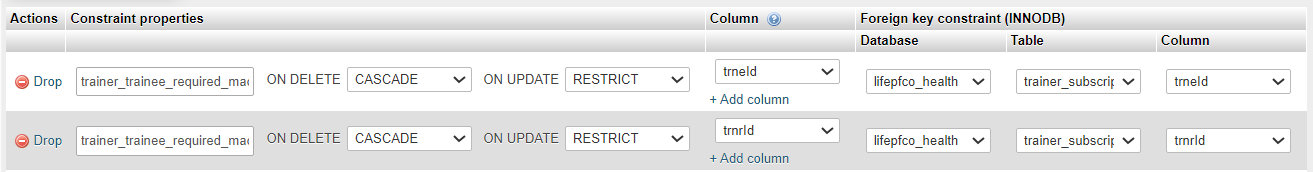
**2-25 : trainer\_trainee\_payments table:**

****

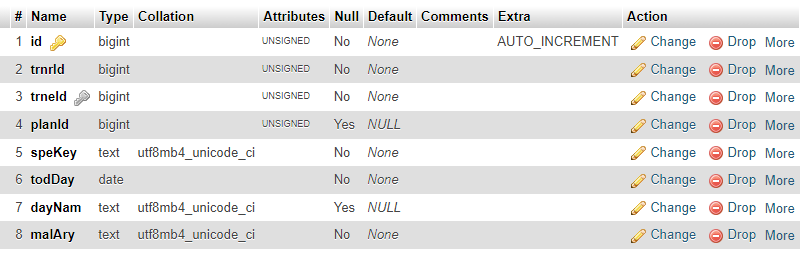
****

**2-26 : trainer\_trainee\_required\_macros table:**

****

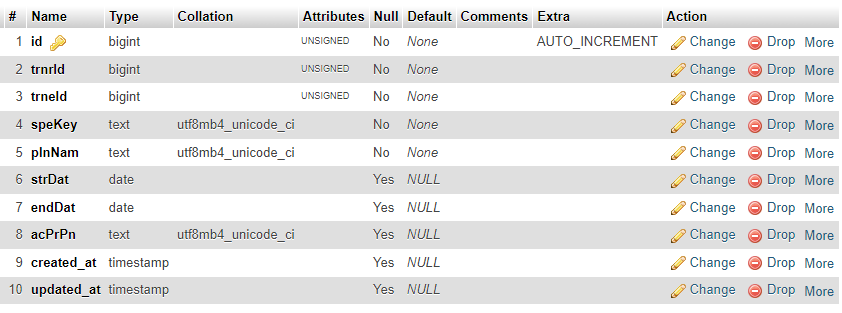
****

**2-27 : trainer\_trainee\_today\_meals table:**

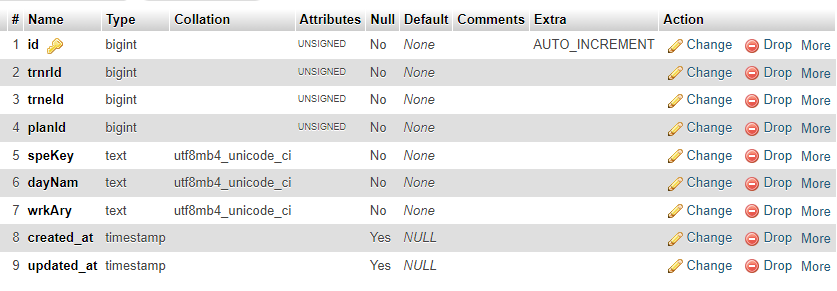
****

****

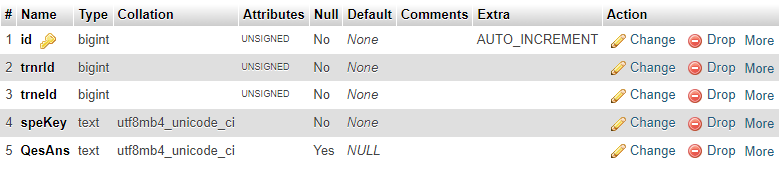
**2-28 : trainer\_trainee\_workouts\_plans table:**

****

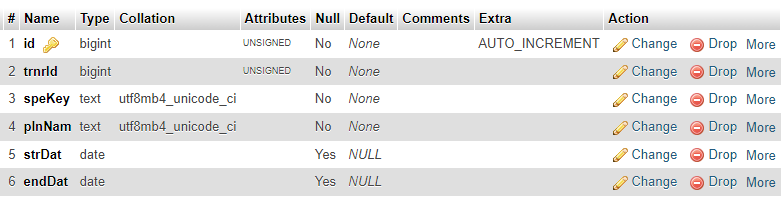
**2-29 : trainer\_trainee\_workouts\_plan\_days table:**

****

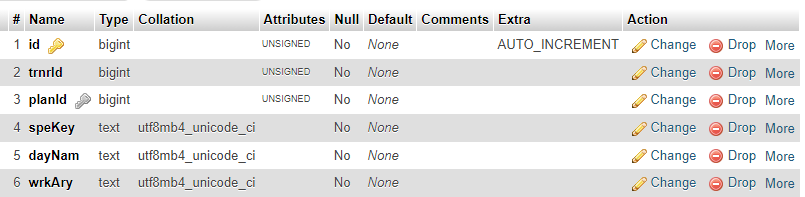
**2-30 : trainee\_\_questionnaire\_page table:**

****

**2-31 : trainer\_workouts\_predefined\_plans table:**

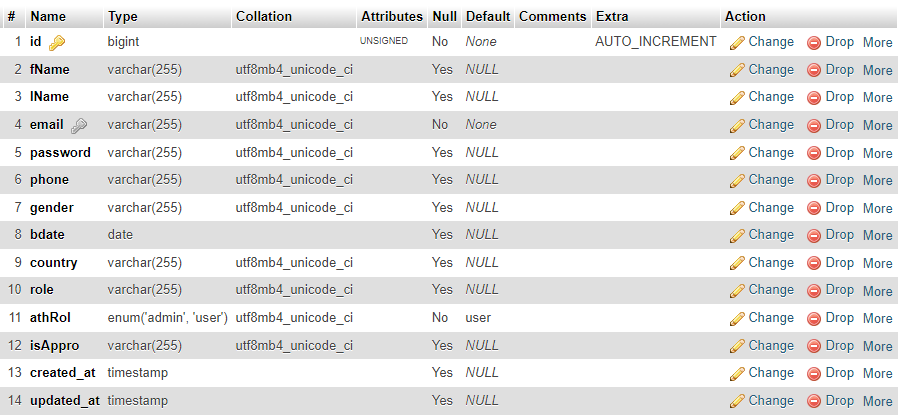
****

**2-32 : trainer\_workouts\_predefined\_plan\_days table:**

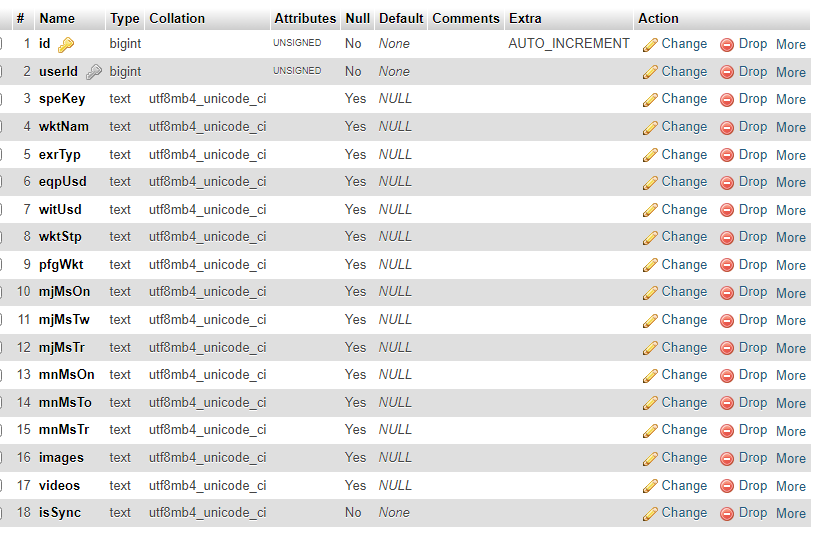
****

****

**2-33 : users table:**

****

**2-34 : workouts table:**

****

****

**Scaling Details:**

Instead of using many rows to insert the user his data I worked on make all the shranking those many rows in one row for many tables that user need to send his data to insert in online database . I used that way in many tables like :

trainer\_manage\_my\_profile, trainer\_meals\_predefined\_plan\_days, trainer\_trainee\_meals\_plan\_days, trainer\_trainee\_today\_meals, trainer\_trainee\_workouts\_plan\_days, trainer\_trainee\_\_questionnaire\_page, trainer\_workouts\_predefined\_plan\_days. You will find that I use in those tables data for many rows in one row in columns like malAry,wrkAry, QesAns, crtfct, socMed you will find that the data in those columns is in arrays stringified to be string until It can fit in one row in one column.

**Security Measures:**

I used Laravel Sanctum in making the Apis authentication and making the tokens that I use in login in the app where I use those token in login and signout in my app . You will find Passwords of the users saved in the database encrypted and saved in the database for savings their data.

**Testing Strategy:**

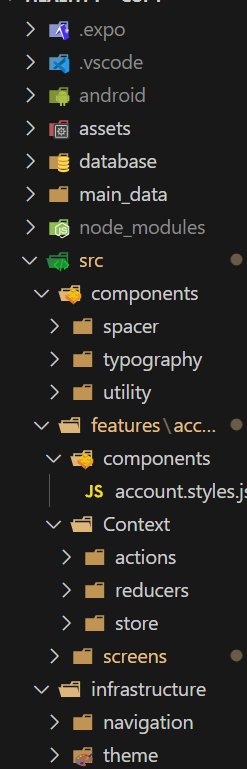
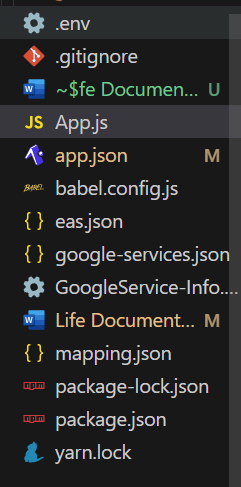
I was sending the data to the backend and check it by checking Laravel.log files if there any errors an check the data reach to the backend to make it fit and suitable when entering into the database.

**Trainer Approval Workflow:**

The trainer make an Account in our app then we send to him an email to validate his email. When the trainer approve on the message we sent to him , his/her account will be registered into the online database and an email will be sent to the admin that a new Trainer has been registered in out database. The admin will open the message pressing on the button in the registration message that button will take him to the phpMyAdmin database dashboard where the admin will change the state of the trainer to approved by changing no in column isAppro to yes in users table . after changing isAppro to yes the trainer account will be appeared in search to the trainee and they can subscribe to him/her.

**2-Front End React Native app Section:**

**1-Frontend Component Hierarchy:**

**1 : android folder :**

**Has the related folders and files that used in building the app in development , debug and Apk versions**

**2 : assets folder:**

**Has the related images, videos, pdf and all the media that used in the app**

**3 : database folder:**

**Has all the functions that used in creating sqllite offline database, tables and inserting data into offline database , getting data from offline database , updating data in offline database , deleting data from offline database.**

**4 : main\_data folder:**

**Has all our workouts,meals json data and functions that we are using to insert into the Sqllite offline database until user can use it offline when ever he wants.**

**5 : node\_modules folder:**

**Has all packages that we installed and used into our app.**

**6 : src folder:**

**Has all big parts of our app in it.**

**6-1 : src/components folder:**

**Has all some parts that we are using in styling some parts in our app.**

**6-2 : src/infrastructure/navigation folder:**

**Has all codes that responsible of navigation stacks in our app. Note: we are using react native navigation in the navigation in our app .**

**6-3-1 : src/infrastructure/navigation folder:**

**Has all codes that responsible of navigation stacks in our app. Note: we are using react native navigation in the navigation in our app .**

**6-3-2 : src/infrastructure/theme folder:**

**Has some files that has some colors,fonts weights, size,sorts,heights that code be used in future in unified the styling (I think they a.**

**6-4-1 : src/features/account/components folder:**

**Has the file that used in styling all the parts in our app .**

**6-4-2 : src\features\account\Context\:**

**Has the files of the authentication data that using context provider to move the data of the authentication through our app.**

**6-4-3 : src\features\account\screens**

**Has the files of our pages in app.**

**6-5 : App.js**

**The main file in our app that has main parts of our app and in it we initialize our offline database tables.**

**6-6 : App.json**

**Has the configuration of our expo app.**

**6-7 : mapping.json**

**File that responsible to update some style manually in ui kitten (examples : dropdown we used in out app we made its background as white and some of its style here).**

**6-8 : package.json**

**File that all the packages that installed in our app using npm.**

**6-9 : yarn.lock**

**File that has the packages that installed in our app using yarn.**

**6-10 : database\synch\_function.js**

**File that has all the functions that used in synch data between offline and online databases.**

**2-Detailed Front End React Native app Functions :**

**2-1 initBodyStatsAndMeasurementsTable():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in creating bodyStatsAndMeasurements table in offline SQLite Database.**

**2-2 insertBodyStatsAndMeasurements():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in inserting or updating the data of body Stats And Measurements into bodyStatsAndMeasurements table in offline SQLite Database.**

**2-3 fetchBodyStatsAndMeasurements():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in getting the data of body Stats And Measurements from bodyStatsAndMeasurements table in offline SQLite Database.**

**2-4 getStatsAndMeasurementsOneRow():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in getting the data of one row of body Stats And Measurements from bodyStatsAndMeasurements table in offline SQLite Database.**

**2-5 getBodyStatsAndMeasurements****UnsyncedRows():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in getting the Unsynced data of body Stats And Measurements from bodyStatsAndMeasurements table in offline SQLite Database. Where we can upload it into online Mysql Database in syncing process.**

**2-6 getBodyStatsAndMeasurementsUnsyncedRows():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in getting the Unsynced data of body Stats And Measurements from bodyStatsAndMeasurements table in offline SQLite Database. Where we can upload it into online Mysql Database in syncing process.**

**2-7 updateStatsAndMeasurementsRowsToSynced():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in convert the Unsynced data of body Stats And Measurements from bodyStatsAndMeasurements table to synced in offline SQLite Database. Where we can’t upload it into online Mysql Database in syncing process any more.**

**2-8 updateStatsAndMeasurementsRowsToSynced():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in convert the Unsynced data of body Stats And Measurements from bodyStatsAndMeasurements table to synced in offline SQLite Database. Where we can’t upload it into online Mysql Database in syncing process any more.**

**2-9 deleteStatsAndMeasurementsUser():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in deleted one row of data of body Stats And Measurements from bodyStatsAndMeasurements table in offline SQLite Database.**

**2-10 fetchbodyStatsAndMeasurementsDateTheSameDateInCalculator ():**

**Its Path :** **database\B\_S\_and\_measurements.js**

**Using:The function that used in getting one row of data of body Stats And Measurements from bodyStatsAndMeasurements table in specific date from offline SQLite Database.**

**2-11 fetchbodyStatsAndMeasurementsLastInsertedRow():**

**Its Path : database\B\_S\_and\_measurements.js**

**Using:The function that used in getting last inserted row of data of body Stats And Measurements from bodyStatsAndMeasurements table from offline SQLite Database.**

**2-12 clearBodyStatsAndMeasurementsTable ():**

**Its Path : database\B\_S\_and\_measurements.js**

**Using:The function that used in clear bodyStatsAndMeasurements table from offline SQLite Database.**

**2-13 deleteBodyStatsAndMeasurementsTable ():**

**Its Path : database\B\_S\_and\_measurements.js**

**Using:The function that used in delete bodyStatsAndMeasurements table from offline SQLite Database.**

**2-14 initCalculatorsTableTable ():**

**Its Path :** **database\calcaulatorsTable.js**

**Using:The function that used in creating calculatorsTable table in offline SQLite Database.**

**2-15 insertCalculatorsTable():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in inserting or updating the data of Calculators into calculatorsTable table in offline SQLite Database.**

**2-16 fetchCalculatorsTable():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in getting the data of Calculators from calculatorsTable table in offline SQLite Database.**

**2-17 fetchCalculatorsTableCalNam():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in getting the data of specific Calculator by its name from calculatorsTable table in offline SQLite Database.**

**2-18 getCalculatorsOneRow():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in getting the data of one row of specific Calculator by its name from calculatorsTable table in offline SQLite Database.**

**2-19 getCalculatorsTableUnsyncedRows ():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in getting the unsynced data of Calculators from calculatorsTable table in offline SQLite Database.**

**2-20 updateCalculatorsRowsToSynced():**

**Its Path :** **database\calcaulatorsTable.js**

**Using:The function that used in convert the Unsynced data of Calculators from calculatorsTable table to synced in offline SQLite Database. Where we can’t upload it into online Mysql Database in syncing process any more.**

**2-21 deleteCalculatorsUser():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in deleting the data of specific Calculator by its name from calculatorsTable table in offline SQLite Database.**

**2-22 fetchCalculatorsTableLastInsertedRow ():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in getting the last row of data of specific Calculator by its name from calculatorsTable table in offline SQLite Database.**

**2-23 clearCalculatorsTableTable():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in clear calculatorsTable table in offline SQLite Database.**

**2-24 deleteCalculatorsTableTable ():**

**Its Path :** **database calcaulatorsTable.js**

**Using:The function that used in delete calculatorsTable table in offline SQLite Database.**

**2-25 initGymEquipmentsTable():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in creating gymEquipments table in offline SQLite Database.**

**2-26 insertGymEquipments():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in inserting the data of Gym Equipments into gymEquipments table in offline SQLite Database.**

**2-27 UpdateGymEquipments():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in updating the data of Gym Equipments into gymEquipments table in offline SQLite Database.**

**2-28 fetchGymEquipments():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in getting the data of Gym Equipments from gymEquipments table in offline SQLite Database.**

**2-29 fetchGymEquipmentsWithoutDeleting():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in getting the data of Gym Equipments that not deleted from gymEquipments table in offline SQLite Database.**

**2-29 getOneGymEquipmentsRow():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in getting one row of the data of Gym Equipments from gymEquipments table in offline SQLite Database.**

**2-30 getGymEquipmentsUnsyncedRows():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in getting Unsynced data of Gym Equipments from gymEquipments table in offline SQLite Database.( Unsynced data : is the data that hasn’t been uploaded into online database)**

**2-31 updateGymEquipmentsRowsToSynced ():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in updatting Unsynced data of Gym Equipments to Synced into gymEquipments table in offline SQLite Database.( Synced data : is the data that hasn been uploaded into online database)**

**2-32 SoftDeleteGymEquipments ():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in deleting data of Gym Equipments in Ui but it still exist in the table in offline SQLite Database.**

**2-33 updateGymEquipmentsName():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in updating gym name in the row of the data of Gym Equipments into the table in offline SQLite Database.**

**2-34 deleteGymEquipmentsRowsWithYes():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in updating gym name in the row of the data of Gym Equipments into the table in offline SQLite Database.**

**2-34 fetchGymEquipmentsLastInsertedRow():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used in getting last inserted row of the data of Gym Equipments from the table in offline SQLite Database.**

**2-35 clearGymEquipmentsTable():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used clearing the gymEquipments table in offline SQLite Database.**

**2-36 deleteGymEquipmentsTable():**

**Its Path :** **database\gym\_equipments\_table.js**

**Using:The function that used deleting the gymEquipments table in offline SQLite Database.**

**2-37 initListOfFoodsTable():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in creating ListOfFoodsTable table in offline SQLite Database.**

**2-38 insertPlansListOfFoods():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in inserting the data of List Of Foods into** **ListOfFoodsTable table in offline SQLite Database.**

**2-39 fetchAlltDaysListOfFoods():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in getting all not deleted foods of the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-40 fetchListOfFoodsWithoutDeleting ():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in getting all not deleted foods in plan of the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-41 checkTodayMeals():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in checking if the users entered meal today or not into ListOfFoodsTable table in offline SQLite Database.**

**2-42 getOneListOfFoodsRow ():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in getting one food of the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-43 getListOfFoods****UnsyncedRows():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in getting Unsynced Rows of the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-44 updateListOfFoodsRowsToSynced():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in update Unsynced Rows of the data of List Of Foods to synced into ListOfFoodsTable table in offline SQLite Database.**

**2-45 updateListOfFoodsRowsToSynced():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in update Unsynced Rows of the data of List Of Foods to synced into ListOfFoodsTable table in offline SQLite Database.**

**2-46 SoftDeleteSelectedMeals():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in deleting data of List Of Foods in Ui but it still exist in the table in offline SQLite Database.**

**2-47 SoftDeleteSelectedMeals():**

**Its Path :** **database\list\_of\_foods.js**

**Using:The function that used in deleting one row of the data of List Of Foods in Ui but it still exist in the table in offline SQLite Database.**

**2-48 updateListOfFoodsName ():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in update name of the plan of the data of List Of Foods into ListOfFoodsTable table in offline SQLite Database.**

**2-49 deleteListOfFoodsRowsWithYes():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in delete the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-50 fetchListOfFoodsLastInsertedRow():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in getting last insert row of the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-51 clearListOfFoodsTable():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in clearing the data of List Of Foods from ListOfFoodsTable table in offline SQLite Database.**

**2-52 deleteListOfFoodsTable ():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in deleting the ListOfFoodsTable table in offline SQLite Database.**

**2-53 deleteListOfFoodsTable ():**

**Its Path : database\list\_of\_foods.js**

**Using:The function that used in deleting the ListOfFoodsTable table in offline SQLite Database.**

**2-54 initListOfFoodsTable():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in creating predefinedMealsTable table in offline SQLite Database.**

**2-55 insertPlansPredefinedMeals():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in inserting our meals into predefinedMealsTable table in offline SQLite Database.**

**2-56 addMainMealsRowsToDatabase ():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in inserting many meals of our meals into predefinedMealsTable table in offline SQLite Database.**

**2-57 fetchLastDayPredefinedMeals ():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting the last inserted meal from predefinedMealsTable table in offline SQLite Database.**

**2-58 fetchAlltDaysPredefinedMeals():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting the all inserted meals from predefinedMealsTable table in offline SQLite Database.**

**2-59 fetchAlltDaysPredefinedMeals():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting the all inserted meals from predefinedMealsTable table in offline SQLite Database.**

**2-60 fetchPredefinedMealsWithoutDeleting():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting the all inserted meals and not deleted in from ui from predefinedMealsTable table in offline SQLite Database.**

**2-61 checkTodayMeals():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in checking if we inserted meal today into predefinedMealsTable table in offline SQLite Database.**

**2-62 getOnePredefinedMealsRow ():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting one row of the data from predefinedMealsTable table in offline SQLite Database.**

**2-63 getPredefinedMealsUnsyncedRows():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in getting Unsynced Rows of the data from predefinedMealsTable table in offline SQLite Database.**

**2-64 updatePredefinedMealsRowsToSynced ():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in updating Unsynced Rows of the data to Synced from predefinedMealsTable table in offline SQLite Database.**

**2-65 SoftDeleteSelectedMeals ():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in deleting the meals just from ui but they are still in the predefinedMealsTable table in offline SQLite Database.**

**2-66 SoftDeletePublicMealsPlanAllDays():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in deleting the meals in plan just from ui but they are still in the predefinedMealsTable table in offline SQLite Database.**

**2-67 updatePredefinedMealsName():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in updating name of the plan of the meals i the predefinedMealsTable table into offline SQLite Database.**

**2-68 deletePredefinedMealsRowsWithYes():**

**Its Path :** **database\predefined\_meals.js**

**Using:The function that used in deleting our meals from the predefinedMealsTable table into offline SQLite Database.**

**2-69 clearPredefinedMealsTable():**

**Its Path : database\predefined\_meals.js**

**Using:The function that used in clearing the data in the predefinedMealsTable table into offline SQLite Database.**

**2-70 deletePredefinedMealsTable ():**

**Its Path : database\predefined\_meals.js**

**Using:The function that used in deleting the predefinedMealsTable table in offline SQLite Database.**

**2-71 initPublicWorkoutsPlanDaysTable():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in creating publicWorkoutsPlanDays table in offline SQLite Database.**

**2-72 insertPlansPublicWorkoutsPlanDays():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in inserting and updating plans sessions into publicWorkoutsPlanDays table in offline SQLite Database.**

**2-73 restartWorkoutByChangingSpekey():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in restart plans sessions by changing spekey in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-74 restartWorkoutByChangingToActive():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in restart plan session to active in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-75 restartAllWorkoutsToActive():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in restart plan all sessions to active in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-76 updateWorkoutByChangingStatusToSkippedOrDone ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in changing plan session statue to Skipped Or Donein publicWorkoutsPlanDays table in offline SQLite Database.**

**2-77 restartAllWorkoutByChangingSpekey ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in restart all plan sessions in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-78 fetchPublicWorkoutsPlanDays():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting all plan sessions from publicWorkoutsPlanDays table in offline SQLite Database.**

**2-79 fetchPublicWorkoutsPlanDaysWithoutDeleting():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting all plan sessions without deleting from publicWorkoutsPlanDays table in offline SQLite Database.**

**2-80 fetchPublicWorkoutsPlanDaysWithoutDeletingWithoutPlnKey():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting all plans sessions without deleting from publicWorkoutsPlanDays table in offline SQLite Database.**

**2-81 getOnePublicWorkoutsPlanDaysRow ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting one plan session from publicWorkoutsPlanDays table in offline SQLite Database.**

**2-82 getPublicWorkoutsPlanDaysUnsyncedRows ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting plans Unsynced sessions from publicWorkoutsPlanDays table in offline SQLite Database.**

**2-83 updatePublicWorkoutsPlanDaysRowsToSynced ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in update plans Unsynced sessions to synced in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-84 SoftDeleteAllPublicWorkoutsPlanDayWorkouts():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in delete plan session in ui but not in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-85 SoftDeletePublicWorkoutsPlanAllDays ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in delete plan all sessions in ui but not in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-86 deletePublicWorkoutsPlanDaysRowsWithYes ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in delete plan all sessions in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-87 fetchPublicWorkoutsPlanDaysLastInsertedRow ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in getting last inserted session in all plans in publicWorkoutsPlanDays table in offline SQLite Database.**

**2-88 clearPublicWorkoutsPlanDaysTable ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in clearing publicWorkoutsPlanDays table in offline SQLite Database.**

**2-89 deletePublicWorkoutsPlanDaysTable ():**

**Its Path :** **database\public\_workouts\_plan\_days.js**

**Using:The function that used in delete publicWorkoutsPlanDays table in offline SQLite Database.**

**2-90 deletePublicWorkoutsPlanDaysTable ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in delete publicWorkoutsPlans table in offline SQLite Database.**

**2-91 initPublicWorkoutsPlansTable():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in creating publicWorkoutsPlans table in offline SQLite Database.**

**2-92 insertPublicWorkoutsPlans ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in inserting plans into publicWorkoutsPlans table in offline SQLite Database.**

**2-93 insertUnlimitedPlansPublicWorkoutsPlans():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in inserting Unlimited plans into publicWorkoutsPlans table in offline SQLite Database.**

**2-94 updatePublicWorkoutsPlans ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in updating plans into publicWorkoutsPlans table in offline SQLite Database.**

**2-95 fetchPublicWorkoutsPlans ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in get Plans from publicWorkoutsPlans table in offline SQLite Database.**

**2-96 fetchPublicWorkoutsPlansWithoutDeleting ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in get Plans without deleted plans from ui from publicWorkoutsPlans table in offline SQLite Database.**

**2-97 getOnePublicWorkoutsPlansRow():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in getting one Plan from publicWorkoutsPlans table in offline SQLite Database.**

**2-98 getPublicWorkoutsPlansUnsyncedRows ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in getting Unsynced Rows Plans from publicWorkoutsPlans table in offline SQLite Database.**

**2-99 updatePublicWorkoutsPlansRowsToSynced():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in updating Unsynced Rows Plans to synced from publicWorkoutsPlans table in offline SQLite Database.**

**2-100 SoftDeletePublicWorkoutsPlans ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in deleting Plan from ui not from publicWorkoutsPlans table in offline SQLite Database.**

**2-101 deletePublicWorkoutsPlansRowsWithYes():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in deleting Plans from publicWorkoutsPlans table in offline SQLite Database.**

**2-102 fetchPublicWorkoutsPlansLastInsertedRow():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in getting last inserted Plans from publicWorkoutsPlans table in offline SQLite Database.**

**2-103 clearPublicWorkoutsPlansTable ():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in clear the data from publicWorkoutsPlans table in offline SQLite Database.**

**2-104 deletePublicWorkoutsPlansTable():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in deleting the data from publicWorkoutsPlans table in offline SQLite Database.**

**2-105 deletePublicWorkoutsPlansTable():**

**Its Path :** **database\public\_workouts\_plans.js**

**Using:The function that used in deleting the data from publicWorkoutsPlans table in offline SQLite Database.**

**2-106 initPublicWorkoutsPlansTable():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in creating startWorkoutTable table in offline SQLite Database.**

**2-107 insertPlansStartWorkout ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in inserting performed sessions into startWorkoutTable table in offline SQLite Database.**

**2-108 fetchLastDayStartWorkouts():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting last performed session from plan from startWorkoutTable table in offline SQLite Database.**

**2-109 fetchLastDayStartWorkoutsWithoutPlnKeyAndDayKey():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting last performed session from all plans startWorkoutTable table in offline SQLite Database.**

**2-110 fetchLastDayStartWorkoutsWithoutPlnKeyAndDayKey():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting last performed session from all plans startWorkoutTable table in offline SQLite Database.**

**2-111 fetchAlltDaysStartWorkouts ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting all performed session from all plans startWorkoutTable table in offline SQLite Database.**

**2-112 fetchStartWorkoutWithoutDeleting ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting all performed sessions without deleted sessions from plan from startWorkoutTable table in offline SQLite Database.**

**2-113 fetchStartWorkoutWithoutDeleting ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting all performed sessions without deleted sessions from plan from startWorkoutTable table in offline SQLite Database.**

**2-114 checkTodayWorkouts ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in checking if the user performed any sessons today in startWorkoutTable table in offline SQLite Database.**

**2-115 checkTodayWorkouts ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in checking if the user performed any sessons today in startWorkoutTable table in offline SQLite Database.**

**2-116 getStartWorkoutUnsyncedRows ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting Unsynced performed sessions from startWorkoutTable table in offline SQLite Database.**

**2-117 updateStartWorkoutRowsToSynced():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in updating Unsynced performed sessions to synced from startWorkoutTable table in offline SQLite Database.**

**2-118 updateStartWorkoutRowsToSynced():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in updating Unsynced performed sessions to synced from startWorkoutTable table in offline SQLite Database.**

**2-119 deleteStartWorkoutRowsWithYes ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in deleting performed sessions from startWorkoutTable table in offline SQLite Database.**

**2-120 deleteStartWorkoutRowsWithYes ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in deleting performed sessions from startWorkoutTable table in offline SQLite Database.**

**2-121 fetchStartWorkoutLastInsertedRow ():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in getting last inserted performed sessions from startWorkoutTable table in offline SQLite Database.**

**2-122 clearStartWorkoutTable():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in clearing startWorkoutTable table in offline SQLite Database.**

**2-123 deleteStartWorkoutTable():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in deleting startWorkoutTable table in offline SQLite Database.**

**2-124 deleteStartWorkoutTable():**

**Its Path :** **database\start\_workout\_db.js**

**Using:The function that used in deleting startWorkoutTable table in offline SQLite Database.**

**2-125 initTargetStatsTable():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in creating targetStats table in offline SQLite Database.**

**2-126 insertOrUpdateTargetStats ():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in inserting and updating data into targetStats table in offline SQLite Database.**

**2-127 fetch****TargetStats():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in getting Target Stats data from targetStats table from offline SQLite Database.**

**2-128 getOneTartgetStatsRow ():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in getting one row from Target Stats data from targetStats table from offline SQLite Database.**

**2-129 getTargetStatsUnsyncedRows ():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in getting Unsynced Rows data from Target Stats data from targetStats table from offline SQLite Database.**

**2-130 updateTargetStatsRowsToSynced ():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in updating Unsynced Rows data To Synced into Target Stats data from targetStats table from offline SQLite Database.**

**2-131 deleteTargetStatsRow():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in deleting one row from data from Target Stats data from targetStats table from offline SQLite Database.**

**2-132 fetchTargetStatsLastInsertedRow():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in getting last inserted row from Target Stats data from targetStats table from offline SQLite Database.**

**2-133 clearTargetStatsTable():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in clearing Target Stats data from targetStats table from offline SQLite Database.**

**2-134 deleteTargetStatsTable():**

**Its Path :** **database\target\_stats.js**

**Using:The function that used in deleting targetStats table from offline SQLite Database.**

**2-135 initTodayMealsTable():**

**Its Path :** **database\today\_meals.js**

**Using:The function that used in creating TodayMealsTable table in offline SQLite Database.**

**2-136 insertOrUpdateTargetStats ():**

**Its Path :** **database\today\_meals.js**

**Using:The function that used in inserting and updating data into TodayMealsTable table in offline SQLite Database.**

**2-137 fetchAlltDaysTodayMeals ():**

**Its Path :** **database\today\_meals.js**

**Using:The function that used in getting data of today from TodayMealsTable table in offline SQLite Database.**

**2-138 fetchTodayMealForCalendar():**

**Its Path :** **database\today\_meals.js**

**Using:The function that used in getting data for spec date from TodayMealsTable table in offline SQLite Database.**

**2-139 fetchAlltDaysTodayMeals ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in getting all data from TodayMealsTable table in offline SQLite Database.**

**2-140 checkTodayMeals():**

**Its Path : database\today\_meals.js**

**Using:The function that used in check if there are Meals for today's date in TodayMealsTable table in offline SQLite Database.**

**2-141 getOneTodayMealsRow():**

**Its Path : database\today\_meals.js**

**Using:The function that used in getting one row data from TodayMealsTable table in offline SQLite Database.**

**2-142 getTodayMealsUnsyncedRows ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in getting Unsynced Rows data from TodayMealsTable table in offline SQLite Database.**

**2-143 updateTodayMealsRowsToSynced():**

**Its Path : database\today\_meals.js**

**Using:The function that used in updatting Unsynced Rows data To Synced into TodayMealsTable table in offline SQLite Database.**

**2-144 SoftDeleteTodayMeal ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in deleting one meal from today meals data from UI not from TodayMealsTable table it still exists in the table in offline SQLite Database.**

**2-145 SoftDeletePublicMealsPlanAllDays():**

**Its Path : database\today\_meals.js**

**Using:The function that used in deleting all meals from today meals data plan from UI not from TodayMealsTable table it still exists in the table in offline SQLite Database.**

**2-146 deleteTodayMealsRowsWithYes ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in deleting all meals from today meals data that deleted from UI to be deleted also from TodayMealsTable table .now it will not exists in the table in offline SQLite Database.**

**2-147 fetchTodayMealsLastInsertedRow ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in getting last inserted row from TodayMealsTable table from offline SQLite Database.**

**2-148 clearTodayMealsTable():**

**Its Path : database\today\_meals.js**

**Using:The function that used in clearing TodayMealsTable table from offline SQLite Database.**

**2-149 deleteTodayMealsTable ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in deleting TodayMealsTable table from offline SQLite Database.**

**2-149 deleteTodayMealsTable ():**

**Its Path : database\today\_meals.js**

**Using:The function that used in deleting TodayMealsTable table from offline SQLite Database.**

**2-150 initTodayMealsTable():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in creating tokens table in offline SQLite Database.**

**2-151 insertOrUpdateTargetStats ():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in inserting and updating tokens data into tokens table in offline SQLite Database.**

**2-152 fetchTokens():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in getting tokens data from tokens table in offline SQLite Database.**

**2-153 getOneToken():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in getting one token from tokens table in offline SQLite Database.**

**2-154 getUserIdFromTokenId ():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in getting user id from token from tokens table in offline SQLite Database.**

**2-155 deleteToken():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in deleting one token from tokens table in offline SQLite Database.**

**2-156 dropTokensTable ():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in deleting tokens table in offline SQLite Database.**

**2-157 dropTokensTable ():**

**Its Path :** **database\tokensTable.js**

**Using:The function that used in deleting tokens table in offline SQLite Database.**

**2-158 initTrainerManageMyProfile ():**

**Its Path :** **database\trainer\_manage\_my\_profile.js**

**Using:The function that used in creating TrainerManageMyProfile table in offline SQLite Database.**

**2-159 insertTrainerManageMyProfile():**

**Its Path :** **database\trainer\_manage\_my\_profile.js**

**Using:The function that used in inserting and updating trainer’s data into TrainerManageMyProfile table in offline SQLite Database.**

**2-160 fetchTrainerManageMyProfile():**

**Its Path :** **database\trainer\_manage\_my\_profile.js**

**Using:The function that used in getting trainer’s data from TrainerManageMyProfile table in offline SQLite Database.**

**2-161 initTrainerPricingCurrency ():**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in creating TrainerPricingCurrency table in offline SQLite Database.**

**2-162 insertTrainerPricingCurrency():**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in inserting and updating trainer pricing’s data into TrainerPricingCurrencytable in offline SQLite Database.**

**2-163 fetchTrainerPricingCurrency():**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in getting trainer pricing’s data from TrainerPricingCurrencytable in offline SQLite Database.**

**2-163 fetchTrainerPricingCurrency():**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in getting trainer pricing’s data from TrainerPricingCurrencytable in offline SQLite Database.**

**2-163 getTrainerPricingCurrencyUnsyncedRows ():**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in getting trainer pricing’s data from TrainerPricingCurrencytable in offline SQLite Database.**

**2-164 function updateTrainerPricingCurrencyRowsToSynced**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in getting trainer pricing’s data from TrainerPricingCurrencytable in offline SQLite Database.**

**2-164 function clearTrainerPricingCurrency()**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in clear TrainerPricingCurrency table in offline SQLite Database.**

**2-165 function deleteTrainerPricingCurrency ()**

**Its Path :** **database\trainer\_pricing\_currency.js**

**Using:The function that used in deleting TrainerPricingCurrency table in offline SQLite Database.**