

משרד התעשיה המסחר והתעסוקה

מה"ט המכון הממשלתי להכשרה בטכנולוגיה ובמדע.

#### 

# 

## 

## המגמה: תוכנה ס'

פרויקט גמר

הנושא: אפליקציה תזונה ואימון

##### **המגיש/ה: אבראהים אבו שאח­­ ­**

##### **ת.ז.:\_315052332**

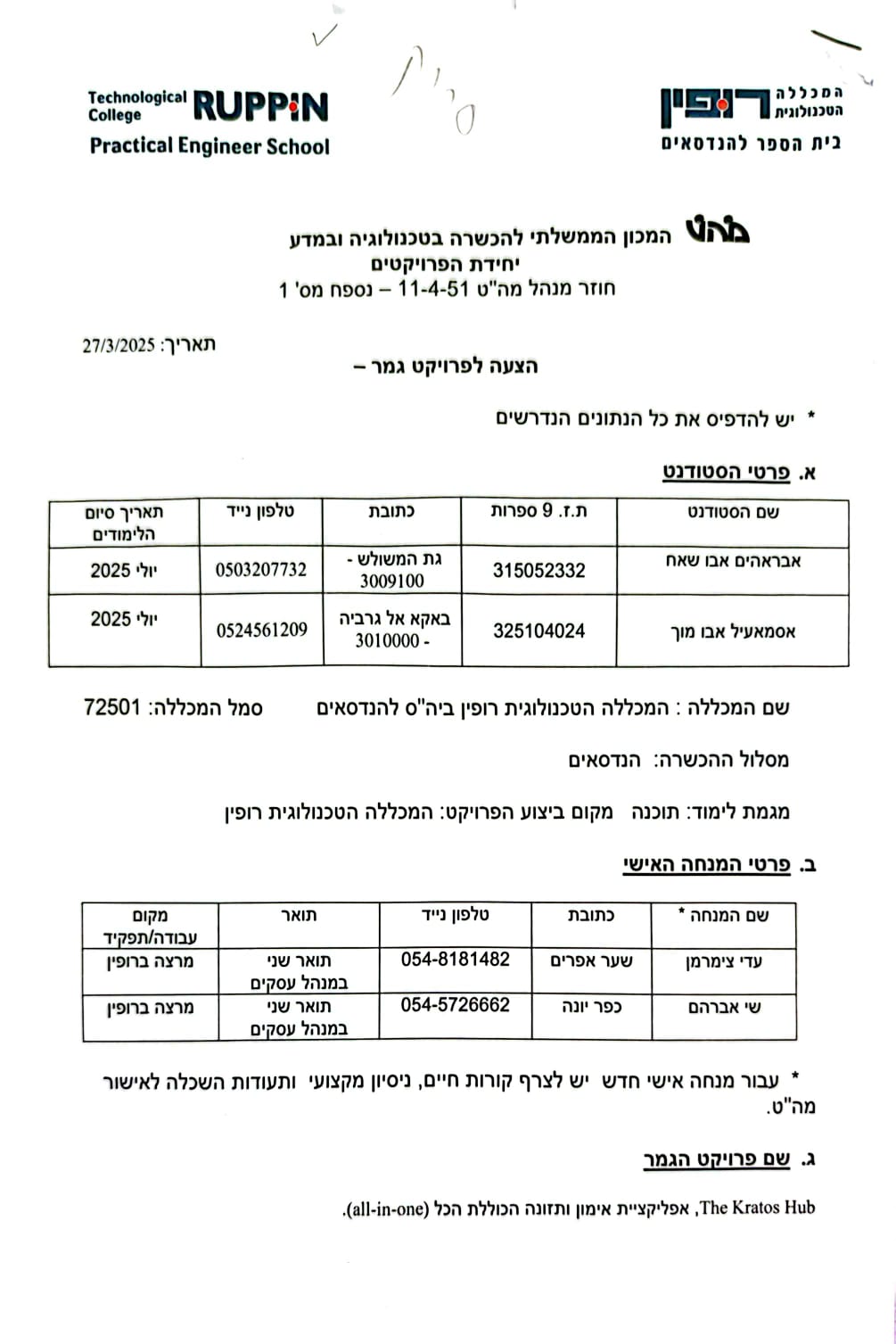
##### **המגיש/ה: ­­­ אסמאעיל אבו מוך­­**

##### **ת.ז.:\_325104024**

**המנחה: עדי צימרמן**

**המנחה: שי אברהם**

מקום לטפסים



Close-up of a document with writing

AI-generated content may be incorrect.

Close-up of a form with writing

AI-generated content may be incorrect.

Close-up of a document with signature

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

1. **שם הפרויקט**

The Kratos Hub, an all-in-one training and nutrition app.

1. **רקע**

**2.1. תיאור ורקע כללי**

Gym culture is a relatively new industry that has gained a lot of popularity over the past 15 to 20 years. It is closely linked to health, fitness, strength, discipline and self-improvement. It also refers to the environment within fitness centers where individuals come together to train, improve their physical health, and socialize. It is often characterized by a sense of community, motivation, and self-improvement. People in gym culture engage in a variety of exercises, from cardio to weightlifting, to achieve their fitness goals.

The name The Kratos Hub means a hub (a gathering place) where athletes gather from all over the world to help each other thrive in their own respective fields. The word Kratos is the name of the god Kratos from Greek mythology. Kratos is the god of strength and power, making the use of his name most appropriate.

Our app will be a tool for those who want to improve their strength and fitness whether for personal or competitive goals.

**2.2. מטרות המערכת**

* Programs will already be built in the app, and the app will wield the user with the ability to create their own programs. These programs can be nutrition programs or training programs.
* The Kratos Hub will also feature a community-based hub where all users can share their programs so that other athletes can use them and create a thriving environment.
* An around-the-clock system that will keep track of the user’s current program to keep reminding them of their daily needs such as taking their meals, drinking enough water throughout the day and more.
* Various activities and exercises will be featured in the app providing the user with the ability to easily create and modify their programs.
* A tool for creating different meals and recipes based on food choices available in the app.
* Statistical analysis for the user to track their progress throughout periods of time.

1. **המצב בשוק והבעיות**
   1. **סקירת מצב קיים בשוק**

**Macros (Nutrition App)**



**Positives:**

* **Simple UI:** Macros has a simple layout design, and users can easily navigate around the app without a lot of clutter and complications.
* **Functionality:** Macros accurately calculates the macros for each meal (protein, fats and carbs) you import into it, which makes it easy for athletes to track what they eat throughout the day. Macros also allows the user to import their weight, goal weight, activity levels and it calculates their basal metabolic rate, BMI and many more.

**Negatives:**

* **Boring:** Macros provides the user with measurement points such as water, BMI and activity levels. But the problem with the app is that it doesn’t have a system that would encourage you to keep track or remind you to have your daily needs and more. It makes using the app rather a hassle.
* **Lack of documentation:** Athletes who like to go back in date and look at their past progress and days are not going to have a good time. The app only features a slider with a left arrow and right arrow in the upper section that you must keep pressing on to reach a certain date or week. It makes it harder for the user to go back and check their meals.

**Alpha Progression Gym Tracker (Workout App)**

A blue and white barbell

Description automatically generated

**Positives:**

* **Categorization:** APGT splits each workout and exercises into groups depending on training style, level and the muscle group they hit to make it simpler for the user to navigate.
* **Animations and pictures:** With each exercise you choose, the app shows you a picture and a video of the exercise to show the user how correctly do them. They also highlight the muscle the exercise hits.

**Negatives:**

* **Locked features:** The app lacks many important features unless the user pays for them.
* **Lack of measurements and personal info inputs:** Unlike Macros, APGT doesn’t ask the user to enter anything other than their weight which takes from its credibility to suggest exercises for each user.
  1. **בעיות במצב הקיים**

We never saw an app that combines nutrition and workouts into one. This is the goal with this app, we want to make an app that coordinates nutrition with gym workouts while also considering the goals of the user.

Another problem is the lack of encouragement and excitement in these apps. Our app will feature milestones and rewards to encourage the user to commit to his progress.

1. **מה הפרויקט אמור לחדש או לשפר**

* Allowing better communication and system that eases sharing nutritional or training data including a chatting system.
* Combining nutrition and gym workouts in one app, so that the user doesn’t have to use multiple apps for both nutrition and workouts.
* Statistical analysis that will help the user easily keep track of progress. With the ability to track based on their time of choosing.
* A system that will constantly remind the user to have your daily meals, meet your daily water intake and more based on the time of the day. This way the user cannot miss their goals.

1. **דרישות מערכת ופונקציונאליות**
   1. **דרישות מערכת**
      1. Functioning database.
      2. An API to inject nutrition and workout information for the system.
      3. Login and registering system for users’ identification.
      4. Platform for users to share ideas (workouts and nutrition).
      5. Platform powerful for customizability of meals and workouts logging.
      6. System responsible for generation of statistical analysis.
      7. Performance requirements:
         1. Fast and responsive interface.
         2. Logging in will not exceed 2 seconds.
         3. Implementation of workouts and nutrition will not exceed 2 seconds.
         4. Profile updating will not exceed 2 seconds.
         5. Profile picture will not exceed 20mb.
         6. Mandatory internet access.
         7. Mandatory database access.
         8. Custom components fit and designed for the application.
         9. Secure backend system that protects user data.
   2. **דרישות פונקציונאליות**
      1. General User:
         1. Login and registration functionality.
         2. Account recovery system.
         3. Nutrition section:
            1. Adjust daily loggings (water intake, etc...).
            2. Manage foods & recipes (with nutritional information).
            3. Manage foods shared to the public nutrition library.
            4. Access to public nutrition library.
            5. View uploader profile page.
         4. Workout section:
            1. Manage workouts.
            2. Share workout to public workout library.
            3. Access to public workout library.
            4. View uploader profile page.
         5. Goals section:
            1. Adjust bodily information (weight, height, etc...).
            2. Adjust physical and activity information (activities, goals, sport, experience, etc...).
            3. Adjust weekly goals (workouts, running, etc...).
            4. Viewing statistics and tracking over periods of time.
         6. Community section:
            1. Ability to search for coaches to hire.
            2. Messaging platform and access to friends/coaches.
            3. Updating residential and locational information.
            4. Access to privacy settings such as hiding user profile.
         7. Profile page:
            1. Adjust personal information.
            2. Upload profile picture.
            3. View messages & notifications.
            4. Adjust measurement units.
            5. Contact us.
            6. Rate us.
      2. Coach (user marked as coach):
         1. Manage clients.
         2. Adjust nutrition & workouts for client.
         3. Send a notification to client.
      3. Timer (system)
         1. Send reminders to athletes for their daily goals (water intake, meals, etc...)
         2. User’s notification management.
         3. Notification in cases of new messages or alerts.
2. **בעיות צפויות במהלך הפיתוח ופתרונות** 
   1. **תיאור הבעיות**

* Keeping up with ever-advantaged technologies.
* Coordinating data between server and client side.
* Connection to the internet, it’s a requirement to be connected to our servers.
* Lag, loading times, flickering and overall app performance.
* Overall security, breach of sensitive information.
* Cross-platform compatibility.
  1. **פתרונות אפשריים**
* **Technological Changes:** Sticking to what we have for now and later in development edit the code and our architecture to fit in with new changes.
* **Data Management:** Managing data exchanged between server-side and client-side is going to be challenging. The solution is to prepare a sheet and sketches to organize how data is handled.
* **Internet Connect:** Almost all applications require internet connectivity. Our application will require the user to have an internet connection, if they don’t, we must handle it and force the user to have internet, or they can’t have access to the app.
* **Performance & User Experience:** A nice-looking, well-designed app can still be a chore to use if it’s slow and unresponsive. Our solution is to prefetch application essential assets before the app runs and leave unnecessary assets to be fetched later when needed.
* **Security:** A deep, multi-layered security system will be built to make sure our user’s information is kept unbreachable.
* **Cross-platform Compatibility:** Many things differ between whatever platform the user might be using. Different devices with different designs and functionality. Our app will be built with that in mind. We will try to make sure every functionality is supported on all platforms.

1. **פתרון טכנולוגי נבחר**
   1. **טופולוגית הפתרון**

Our project will run on a three-layer architecture with client-side, server and database.

* **Client-side:**

Our UI or client-side will run on React Native under Expo support. It will feature support for both IOS & Android machines, a website that serves more as an informational site about the program and for communication & contact information. All client-side programs requests are managed by the server-side.

* **Third-party communicator:**

Communication between the server and both the client and the database will be managed using a third-party server like Somee, Netlify or Vencel. which will require internet connection.

* **Server-side:**

Our server will be the mastermind behind every user’s action. It will handle requests such as fetching and storing data. It will run on Node.js or C# depending on later assessments.

* **Database:**

Our database will contain all the information about all applications related to the project and our users. It will also use the third-party communicator to connect with the server.

* 1. **טכנולוגיות בשימוש**

A variety of technologies will be implemented to provide the most efficient and seamless experience for our clients.

* **Client-side:**

Our client-side applications will incorporate various AI technologies, including third-party models, to provide optimized user experience. These technologies will enable personalized features such as filtering workouts and food selections tailored to each user's specific needs and preferences.

Additionally, we will incorporate geographical and mapping technologies to support location-based services, such as helping athletes find nearby coaches. To ensure optimal matching, filtering coaches based on their proximity to the athlete will be a key feature.

* **Server-side:**

The server is a very crucial part of any project. It keeps a smooth flow of information and so it needs to be optimized for proper communication between the client and the server.

* + **Coding technologies:**
    - **Singleton:**

The use of singleton prevents collision of information when different users send request concurrently.

* + - **Factory Method:**

Object factories allow a smooth experience for us, the programmers, to easily create different objects as our server will have many inherited or composed classes.

* + **Architectural technologies:**
    - **Restful Api:**

The server will perform POST, GET, UPDATE & DELETE functions.

* + - **Middlewares:**

Several middlewares will be installed to ensure fast and efficient responses from the server.

* + - **CORS:**

CORS for access to the restricted data & security.

* + - **Other technologies:**

There are other technologies that might be used and will be inserted here when confirmed.

* 1. **שפות הפיתוח.**
* **Front-end:**

We think full-stack languages with React Native framework is the best fit for our application because it serves as an excellent front-end visual design and functionality and it offers compatibility amongst most devices.

* **Server:**

For the server side, we will use Node.js for its compatibility with react with both being JavaScript languages. And with it we will program our algorithms between the front and back ends.

* **Database:**

For data storage, we will use SQL.

* **APIs:**

For API’s we will need something that offers nutritional data such as World Health Organization’s “GHO OData API” or US Food & Drug Administration API.

* 1. **תיאור הארכיטקטורה הנבח**

The project will run on a three-tier architecture design.

Client side will use React Native.

Server side will use Node.js.

Database will be SQL.

* 1. **חלוקה לתכניות ומודולי**
  2. **סביבת השרת**
* The server will run on Node.js with SQL as its database.
* The server will include all security measures, CORS functionality, JWT Tokens...
* Class based system allowing for better management of backend functionality.
* On run DB and Tables creation if non-existent to prevent server crashes.
* Sensitive information such as passwords will be hashed.
* Data such as text messages will be encrypted with both clients sharing a key for decryption (End-to-End Encryption).
* The server & database will be hosted by Somee and client-side will be hosted by Netlify.

* 1. **ממשק המשתמש/לקוח – GUI**

**INTRODUCTION**

**A screenshot of a phone

AI-generated content may be incorrect.**

**REGISTRATION**

**A screenshot of a phone

AI-generated content may be incorrect.**

****

**HOMEPAGE**

**A screenshot of a phone

AI-generated content may be incorrect.**

**Screenshot of a phone screen

AI-generated content may be incorrect.**

**Screens screenshot of a phone

AI-generated content may be incorrect.**

**PROFILE**

**A screenshot of a phone

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**GOALS & NUTRITION**

**A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**7.8. ממשקים למערכות אחרות/API**

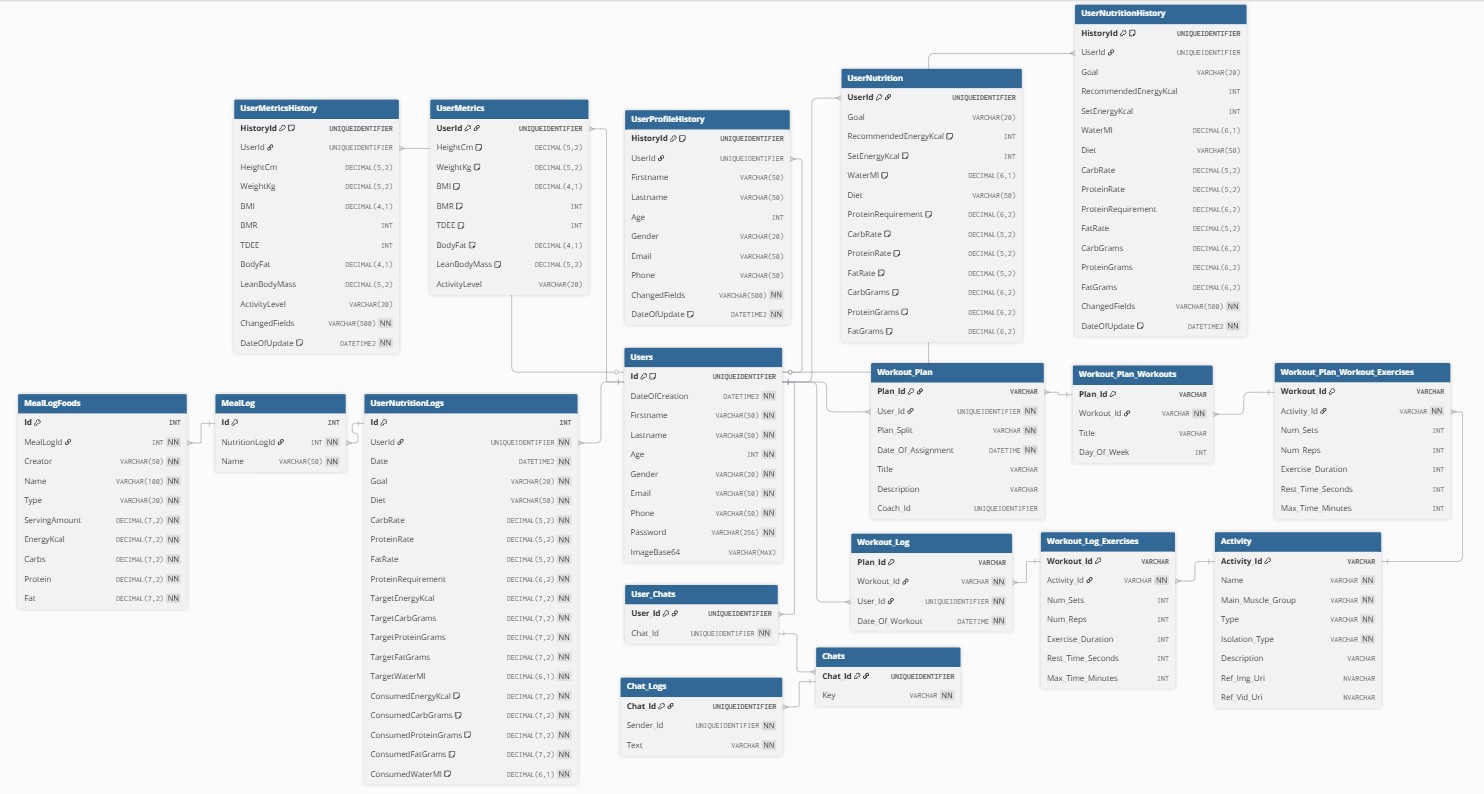
Expo to run our React Native code, different NPM packages for React, and an API for nutritional data.

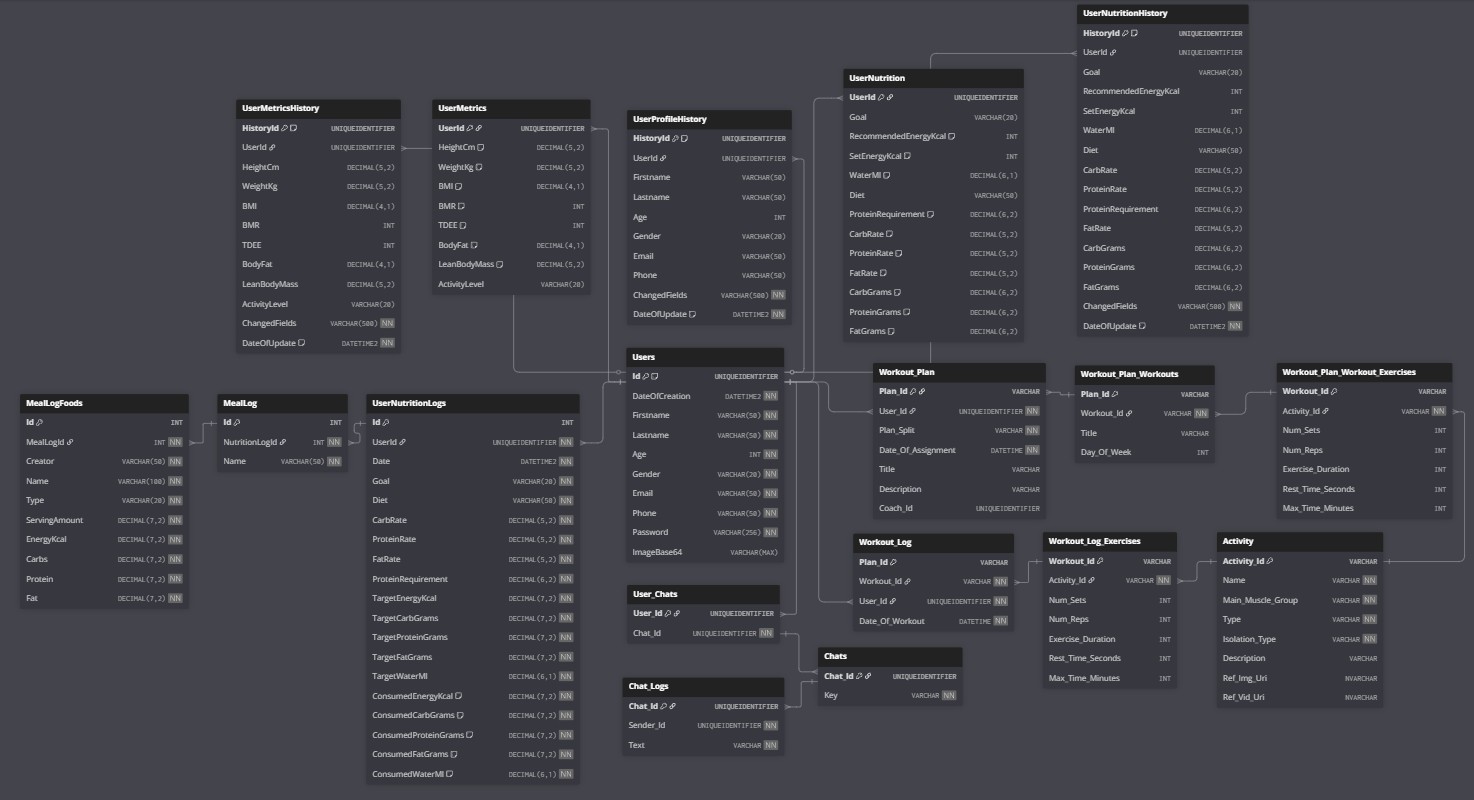
**7.9. שימוש בחבילות תוכנה**

* **Client:**
  + Different packages from Expo like camera & camera permissions management, notification pushing, timer functions, geographical functions and more.
  + Styling packages like Material UI & Bootstrap.
* Server:
  + Express.js for server system.
  + Morgan for additional middleware support.
  + Crypto & Bcrypt for encryption and hashing.

1. **שימוש במבני נתונים וארגון קבצים** 
   1. **מבני הנתונים**

**Database Tables**

****

****

**Table Name: Users**

**Table Description: Stored information about every user in the system.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Id | UNIQUEIDENTIFIER | Primary Key, Default NEWID() | User unique ID |
| DateOfCreation | DATETIME2 | Not Null | Account creation date |
| Firstname | VARCHAR(50) | Not Null | User first name |
| Lastname | VARCHAR(50) | Not Null | User last name |
| Age | INT | Not Null | User age |
| Gender | VARCHAR(20) | Not Null | User gender |
| Email | VARCHAR(50) | Not Null, Unique | User email |
| Phone | VARCHAR(50) | Not Null, Unique | User phone number |
| Password | VARCHAR(256) | Not Null | Hashed password |
| ImageBase64 | VARCHAR(MAX) | NONE | Profile picture in base64 |

**Table Name: UserMetrics**

**Table Description: Storied information about user metrics such as height, weight.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| UserId | UNIQUEIDENTIFIER | Primary Key | Linked user ID |
| HeightCm | DECIMAL(5,2) | Default 0 | User height in cm |
| WeightKg | DECIMAL(5,2) | Default 0 | User weight in kg |
| BMI | DECIMAL(4,1) | Default 0 | Body Mass Index |
| BMR | INT | Default 0 | Basal Metabolic Rate |
| TDEE | INT | Default 0 | Total Daily Energy Expenditure |
| BodyFat | DECIMAL(4,1) | Default 0 | Body fat percentage |
| LeanBodyMass | DECIMAL(5,2) | Default 0 | Lean body mass |
| ActivityLevel | VARCHAR(20) | NOT NULL | Activity level |

**Table Name: UserNutrition**

**Table Description: Storied information about user nutrition such as daily calorie intake.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| UserId | UNIQUEIDENTIFIER | Primary Key | Linked user ID |
| Goal | VARCHAR(20) | NOT NULL | Nutrition goal |
| RecommendedEnergyKcal | INT | Default 0 | Recommended daily calories |
| SetEnergyKcal | INT | Default 0 | Set daily calories |
| WaterMl | DECIMAL(6,1) | Default 0 | Water intake in ml |
| Diet | VARCHAR(50) | None | Diet type |
| ProteinRequirement | DECIMAL(6,2) | Default 0 | Protein requirement |
| CarbRate | DECIMAL(5,2) | Default 0 | Carb percentage |
| ProteinRate | DECIMAL(5,2) | Default 0 | Protein percentage |
| FatRate | DECIMAL(5,2) | Default 0 | Fat percentage |
| CarbGrams | DECIMAL(6,2) | Default 0 | Carbs in grams |
| ProteinGrams | DECIMAL(6,2) | Default 0 | Protein in grams |
| FatGrams | DECIMAL(6,2) | Default 0 | Fat in grams |

**Table Name: UserProfileHistory**

**Table Description: Storied information about user profile over periods of time for statistics.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| HistoryId | UNIQUEIDENTIFIER | Primary Key, Default NEWID() | History record ID |
| UserId | UNIQUEIDENTIFIER | NONE | Linked user ID |
| Firstname | VARCHAR(50) | NONE | Updated first name |
| Lastname | VARCHAR(50) | NONE | Updated last name |
| Age | INT | NONE | Updated age |
| Gender | VARCHAR(20) | NONE | Updated gender |
| Email | VARCHAR(50) | NONE | Updated email |
| Phone | VARCHAR(50) | NONE | Updated phone |
| ChangedFields | VARCHAR(500) | Not Null | Fields that changed |
| DateOfUpdate | DATETIME2 | Not Null, Default SYSUTCDATETIME() | Update timestamp |

**Table Name: UserMetricsHistory**

**Table Description: Storied information about user metrics over periods of time for statistics.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| HistoryId | UNIQUEIDENTIFIER | Primary Key, Default NEWID() | History record ID |
| UserId | UNIQUEIDENTIFIER | NONE | Linked user ID |
| HeightCm | DECIMAL(5,2) | NONE | Updated height cm |
| WeightKg | DECIMAL(5,2) | NONE | Updated weight kg |
| BMI | DECIMAL(4,1) | NONE | Updated BMI |
| BMR | INT | NONE | Updated BMR |
| TDEE | INT | NONE | Updated TDEE |
| BodyFat | DECIMAL(4,1) | NONE | Updated body fat |
| LeanBodyMass | DECIMAL(5,2) | NONE | Updated lean body mass |
| ActivityLevel | VARCHAR(20) | NONE | Updated activity level |
| ChangedFields | VARCHAR(500) | Not Null | Fields that changed |
| DateOfUpdate | DATETIME2 | Not Null, Default SYSUTCDATETIME() | Update timestamp |

**Table Name: UserNutritionHistory**

**Table Description: Storied information about user nutrition over periods of time for statistics.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| HistoryId | UNIQUEIDENTIFIER | Primary Key, Default NEWID() | History record ID |
| UserId | UNIQUEIDENTIFIER |  | Linked user ID |
| Goal | VARCHAR(20) |  | Updated nutrition goal |
| RecommendedEnergyKcal | INT |  | Updated recommended calories |
| SetEnergyKcal | INT |  | Updated set calories |
| WaterMl | DECIMAL(6,1) |  | Updated water intake (ml) |
| Diet | VARCHAR(50) |  | Updated diet type |
| CarbRate | DECIMAL(5,2) |  | Updated carb percentage |
| ProteinRate | DECIMAL(5,2) |  | Updated protein percentage |
| ProteinRequirement | DECIMAL(6,2) |  | Updated protein requirement |
| FatRate | DECIMAL(5,2) |  | Updated fat percentage |
| CarbGrams | DECIMAL(6,2) |  | Updated carbs in grams |
| ProteinGrams | DECIMAL(6,2) |  | Updated protein in grams |
| FatGrams | DECIMAL(6,2) |  | Updated fat in grams |
| ChangedFields | VARCHAR(500) | Not Null | Fields that changed |
| DateOfUpdate | DATETIME2 | Not Null, Default SYSUTCDATETIME() | Update timestamp |

**Table Name: UserNutritionLogs**

**Table Description: Storied information about user nutritional logs over periods of time for them to read, edit and more**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Id | INT | Primary Key, Auto Increment | Log ID |
| UserId | UNIQUEIDENTIFIER | Not Null | Linked user ID |
| Date | DATETIME2 | Not Null | Date of log |
| Goal | VARCHAR(20) | Not Null | Nutrition goal |
| Diet | VARCHAR(50) | Not Null | Diet type |
| CarbRate | DECIMAL(5,2) | Not Null | Carb percentage |
| ProteinRate | DECIMAL(5,2) | Not Null | Protein percentage |
| FatRate | DECIMAL(5,2) | Not Null | Fat percentage |
| ProteinRequirement | DECIMAL(6,2) | Not Null | Protein requirement |
| TargetEnergyKcal | DECIMAL(7,2) | Not Null | Target calories |
| TargetCarbGrams | DECIMAL(7,2) | Not Null | Target carbs in grams |
| TargetProteinGrams | DECIMAL(7,2) | Not Null | Target protein in grams |
| TargetFatGrams | DECIMAL(7,2) | Not Null | Target fat in grams |
| TargetWaterMl | DECIMAL(6,1) | Not Null | Target water in ml |
| ConsumedEnergyKcal | DECIMAL(7,2) | Not Null, Default 0 | Actual calories consumed |
| ConsumedCarbGrams | DECIMAL(7,2) | Not Null, Default 0 | Actual carbs consumed |
| ConsumedProteinGrams | DECIMAL(7,2) | Not Null, Default 0 | Actual protein consumed |
| ConsumedFatGrams | DECIMAL(7,2) | Not Null, Default 0 | Actual fat consumed |
| ConsumedWaterMl | DECIMAL(6,1) | Not Null, Default 0 | Actual water consumed |

**Table Name: MealLog**

**Table Description: Stored meals for each nutrition log.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| **Id** | **INT** | **Primary Key, Auto Increment** | **Meal ID** |
| **NutritionLogId** | **INT** | **Not Null** | **Linked nutrition log ID** |
| **Name** | **VARCHAR(50)** | **Not Null** | **Meal name** |

**Table Name: MealLogFoods**

**Table Description: Stored foods for each meal log.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Id | INT | Primary Key, Auto Increment | Food entry ID |
| MealLogId | INT | Not Null | Linked meal ID |
| Creator | VARCHAR(50) | Not Null | Who logged it |
| Name | VARCHAR(100) | Not Null | Food or drink name |
| Type | VARCHAR(20) | Not Null | 'food' or 'liquid' |
| ServingAmount | DECIMAL(7,2) | Not Null | Amount served |
| EnergyKcal | DECIMAL(7,2) | Not Null | Energy in kcal |
| Carbs | DECIMAL(7,2) | Not Null | Carbs in grams |
| Protein | DECIMAL(7,2) | Not Null | Protein in grams |
| Fat | DECIMAL(7,2) | Not Null | Fat in grams |

**Table Name: UserChats**

**Table Description: A table responsible for coordinating which chats belong to which users.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| User\_Id | UNIQUEIDENTIFIER | Primary Key | Linked user ID |
| Chat\_Id | UNIQUEIDENTIFIER | Not Null | Linked chat ID |

**Table Name: Chats**

**Table Description: Table to store all chats and their key.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Chat\_Id | UNIQUEIDENTIFIER | Primary Key | Chat ID |
| Key | VARCHAR | Not Null | Chat access key |

**Table Name: ChatsLogs**

**Table Description: Table to store all chat logs and who sent the message.**

| **Field Name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Chat\_Id | UNIQUEIDENTIFIER | Primary Key | Linked chat ID |
| Sender\_Id | UNIQUEIDENTIFIER | Not Null | ID of message sender |
| Text | VARCHAR | Not Null | Chat message text |

**Table Name: WorkoutPlan.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Plan’s id in the system | Primary Key, Not Null | Varchar | plan\_id |
| The user’s id this plan belongs to | Not Null | Varchar | user\_id |
| The split this plan follows | None | Varchar | plan\_split |
| The date this plan was created | Not Null | Datetime | date\_of\_assignment |
| The plan’s title | None | Varchar | title |
| A brief description of the plan | None | Varchar | description |
| The coach that assigned this plan (if there’s a coach) | None | Varchar | coach\_id |

**Table Description: Stored every user’s workout plans.**

**Table Name: WorkoutPlanWorkouts.**

**Table Description: Assistant table used to determine which workout logs belong to which plans.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Plan’s id in the system of which this day belongs to | Primary Key, Not Null | Varchar | plan\_id |
| The id of the workout this day belongs to | Not Null | Varchar | workout\_id |
| The workout’s title | None | Varchar | title |
| Which day of the week this workout belongs to | None | Int | day\_of\_week |

**Table Name: WorkoutPlanWorkoutExercises.**

**Table Description: Assistant table to determine which exercises belong to which plans.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Workout’s id which this exercise belongs to | Primary Key, Not Null | Varchar | workout\_id |
| Activity’s id this exercise utilizes | Not Null | Varchar | activity\_id |
| Times the activity to be executed | None | Int | num\_sets |
| Repetitions in each set | None | Int | num\_reps |
| Exercise duration in seconds | None | Int | exercise-duration |
| Rest time between each set | None | Int | rest\_time\_seconds |
| Maximum time this exercise should take | None | Int | max\_time\_minutes |

**Table Name: WorkoutLog.**

**Table Description: Stored daily workout logs for each user.**

**Table Name: WorkoutLogExercises.**

**Table Description: Assistant table to reference exercises used in the workout log.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Log’s id in the system | Primary Key, Not Null | Varchar | log\_id |
| Workout’s id in the system | Not Null | Varchar | workout\_id |
| The id of the athlete this workout log belongs to | Not Null | Varchar | user\_id |
| The date when this workout was logged | Not Null | Datetime | date\_of\_workout |

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Workout’s id which this exercise belongs to | Primary Key, Not Null | Varchar | workout\_id |
| Activity’s id this exercise utilizes | Not Null | Varchar | activity\_id |
| Times the activity to be executed | None | Int | num\_sets |
| Repetitions in each set | None | Int | num\_reps |
| Exercise duration in seconds | None | Int | exercise-duration |
| Rest time between each set | None | Int | rest\_time\_seconds |
| Maximum time this exercise should take | None | Int | max\_time\_minutes |

**Table Name: Activity.**

**Table Description: Stored activities users can utilize in the system.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Characteristics** | **Type** | **Field Name** |
| Activity’s id in the system | Primary Key, Not Null | Varchar | activity\_id |
| The name of the activity | Not Null | Varchar | name |
| The date when this workout was logged | Not Null | varchar | main\_muscle\_group |
| The type of activity (Cardiovascular, hypertrophy, aerobic) | Not Null | varchar | type |
| Compound or isolation movement? | Not Null | varchar | isolation\_type |
| A brief description of the activity | None | varchar | description |
| Reference for image of the activity | None | varchar | ref\_img\_uri |
| Reference for video of the activity | None | varchar | ref\_vid\_uri |

* 1. **שיטת האחסון**

**SQL:**

Tabular database using a Relational Database Management System (RDBMS) such as MySQL or PostgreSQL.  
These systems organize data into related tables using rows and columns.  
The development language used to communicate with the system is SQL (Structured Query Language), an ANSI-standard language for querying, inserting, updating, and deleting data.  
It also supports schema definition, constraints, transactions, and the establishment of relationships through primary and foreign keys. Standard SQL is widely adopted and compatible with many RDBMS platforms.

* 1. **מנגנוני התאוששות**

For recovery, we decided to use a mix of regular backups and replication to make sure our system can recover quickly and stay available if something fails.

**Backup and Restore**

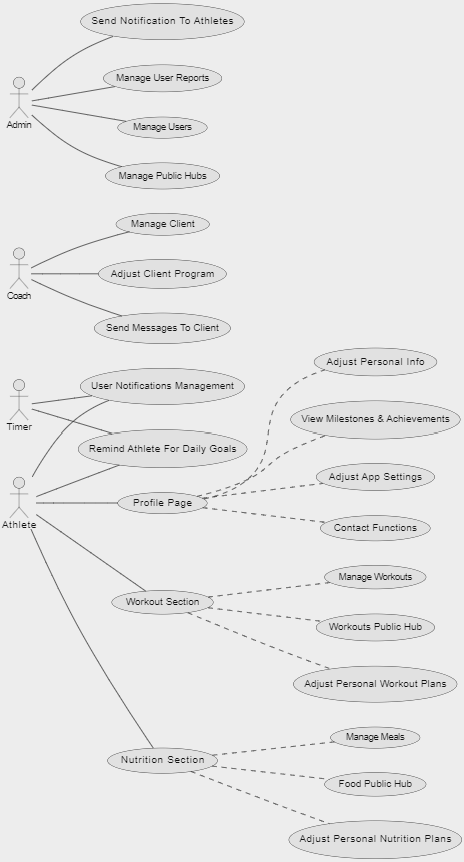
We set up full backups to run on a regular schedule, and in between those, we use differential backups to keep things lighter and faster. We also back up the transaction logs frequently so we can recover the database to a specific point in time if needed. This setup gives us a good balance between reliability and flexibility. The only downside we’ve noticed is that restoring everything especially when combining full, differential, and log backups can take a bit of time, but it’s a trade-off we’re okay with.

**Replication**

To improve availability and take some pressure off the main database, we also implemented replication. In SQL Server, we went with transactional replication since it keeps the data closely in sync, which is important for our use case. On the MongoDB side, we set up a replica set, which automatically keeps multiple copies of the data and handles failover on its own. This has been useful for reporting and read-heavy tasks, which we direct to the secondary servers so the primary can focus on handling updates.

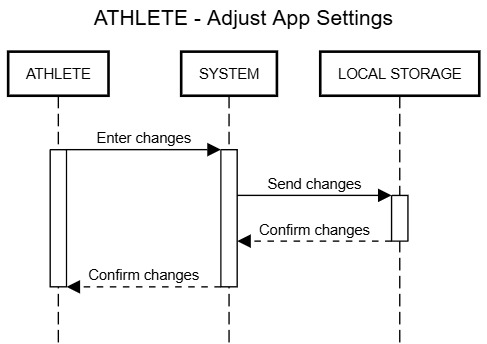
1. **תרשימי מערכת מרכזיים**

**9.1. Use Case**

****

**9.2 Sequence Diagram**

**The process of adjusting app preferences**

****

**The process of the system displaying current achievements**

**A diagram of a system

AI-generated content may be incorrect.**

**The process of adjusting personal information, certain validation measures are included**

**A diagram of a diagram

AI-generated content may be incorrect.**

**The process of managing nutritional plans, a little more complex**

**A diagram of a software project

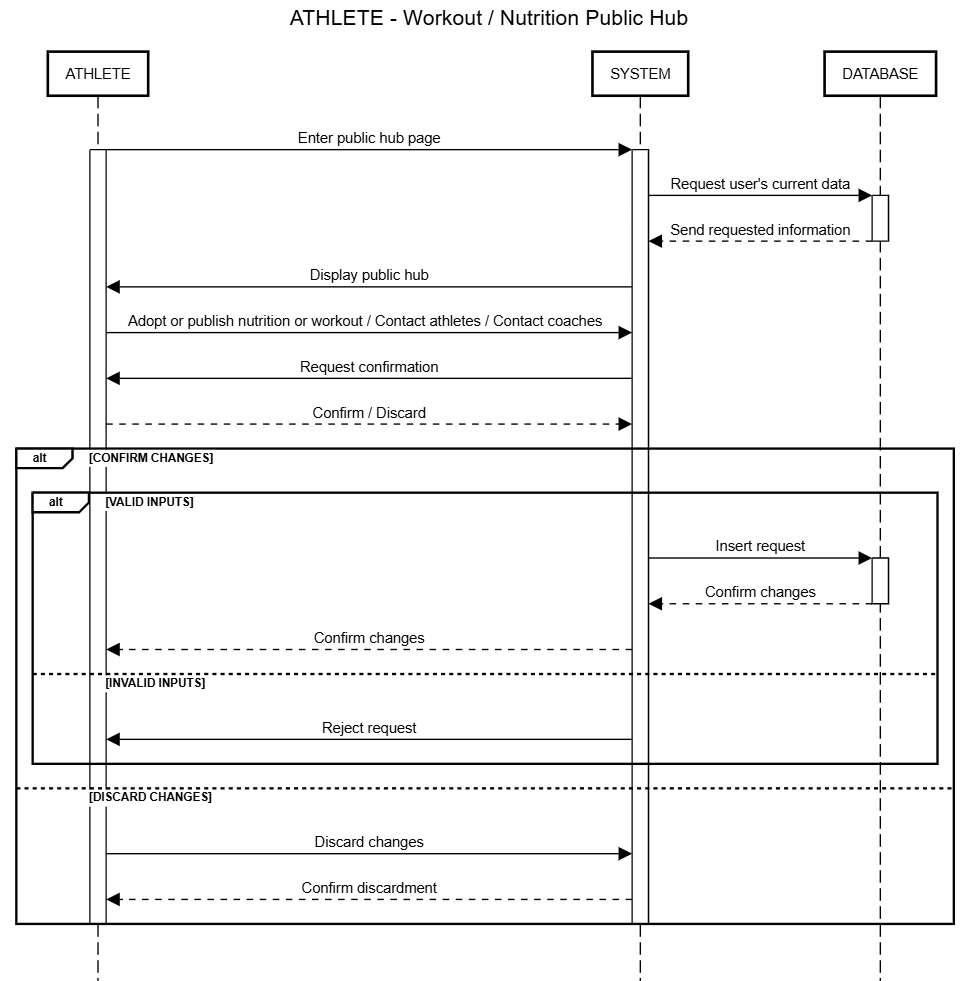
AI-generated content may be incorrect.**

**Same as nutrition**

**A diagram of a software application

AI-generated content may be incorrect.**

**Interaction with public hubs workouts / nutrition**

****

**The process of managing a client as coach, requires client’s approval**

A diagram of a diagram

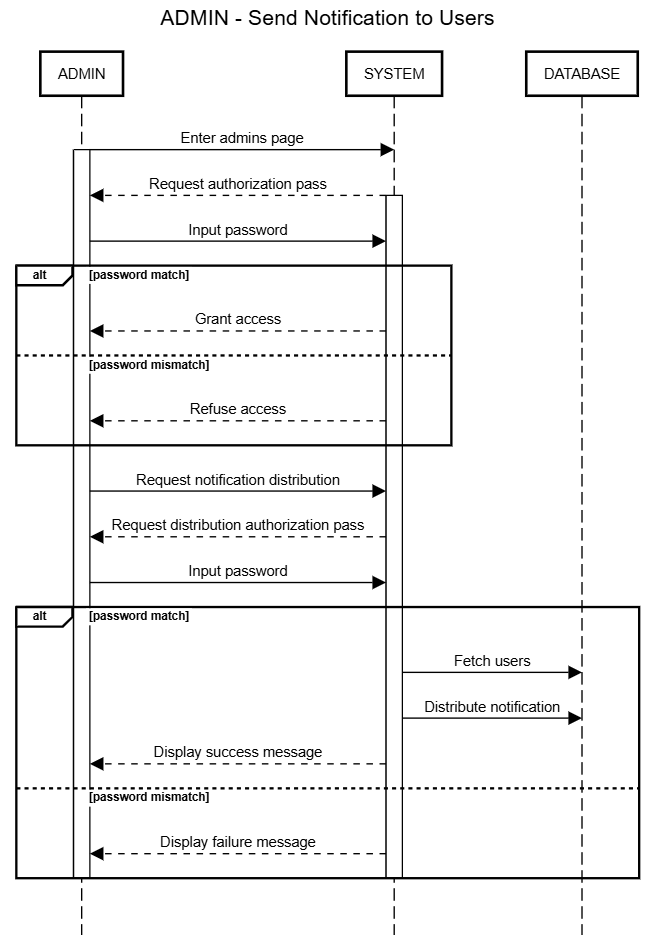
AI-generated content may be incorrect.

**Chatting between users of all kind**

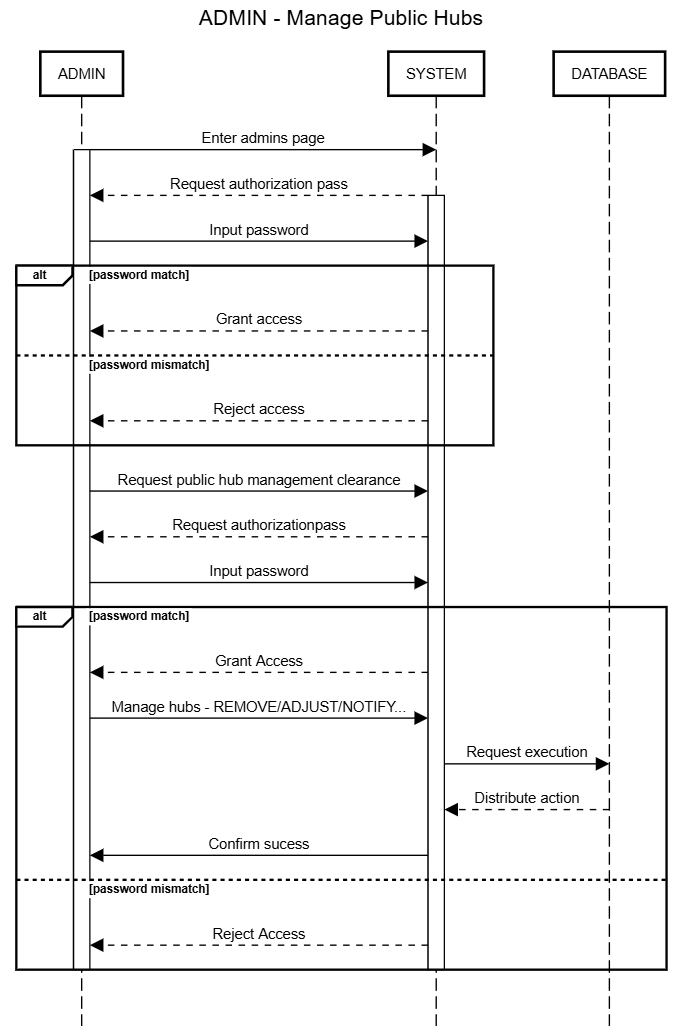
**A diagram of a diagram

AI-generated content may be incorrect.**

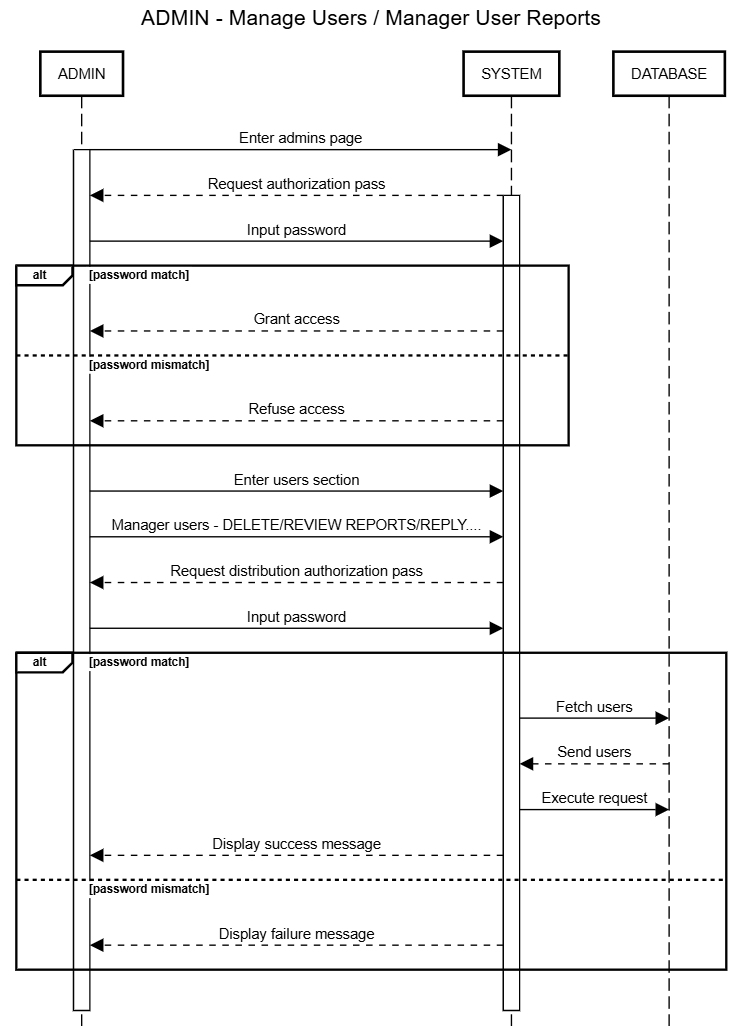
**Sending notifications to users manually as admin**

****

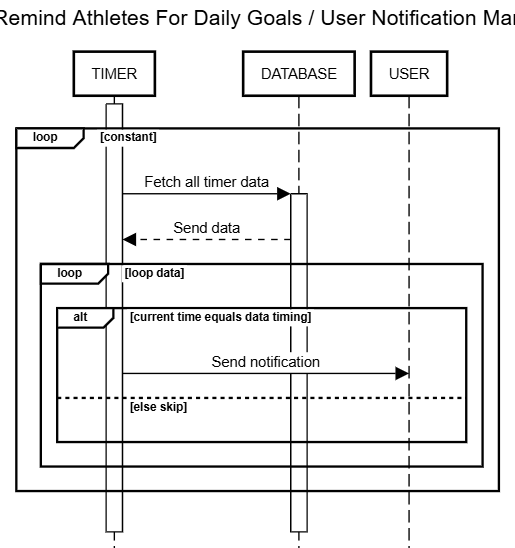
**Moderating public hubs as admin or moderator**

****

**Managing users and their reports**

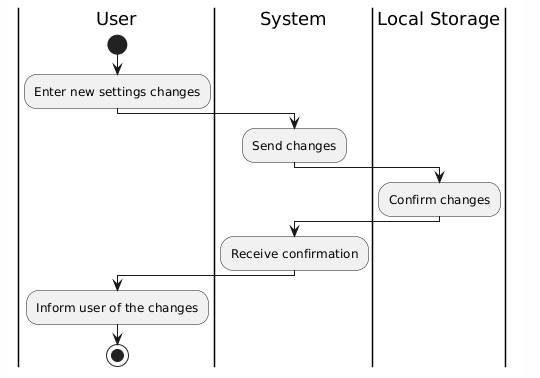
****

**Timer automatically sending notifications to users for updates**

****

**9.3Data Flow**

**Adjust App Settings**

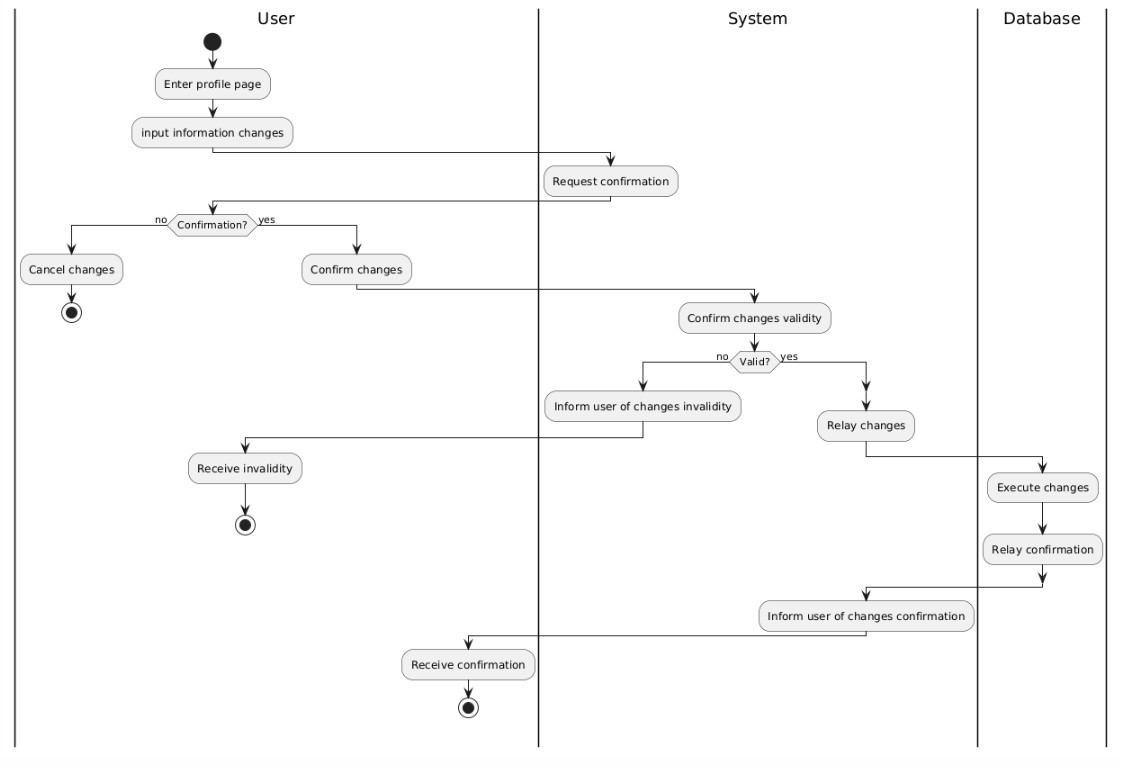


**View Milestones & Achievements**

A diagram of a user

AI-generated content may be incorrect.

**Adjust Personal Info**

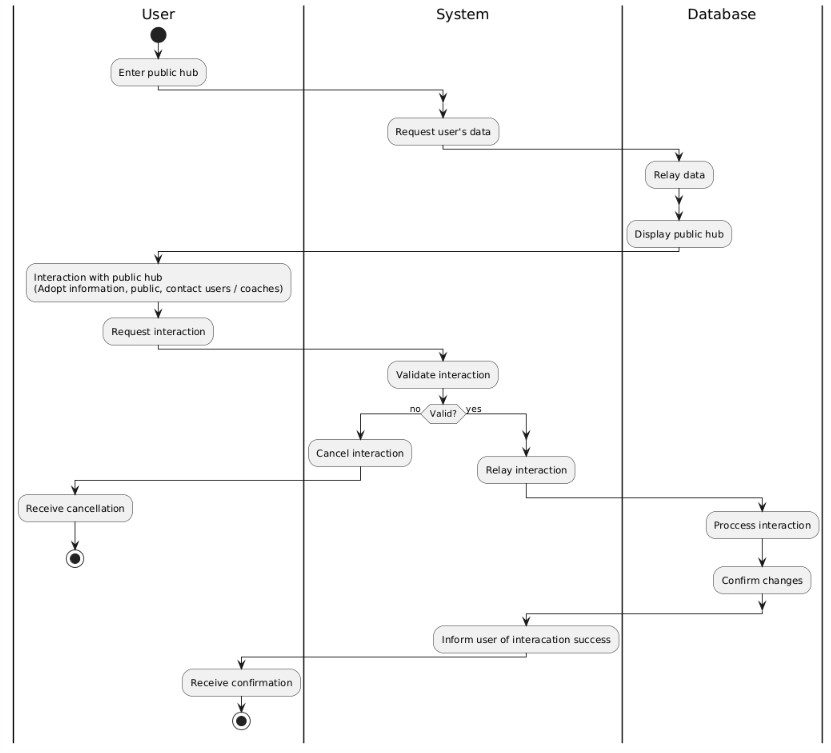


**Manage Nutriton & Workout Plans**

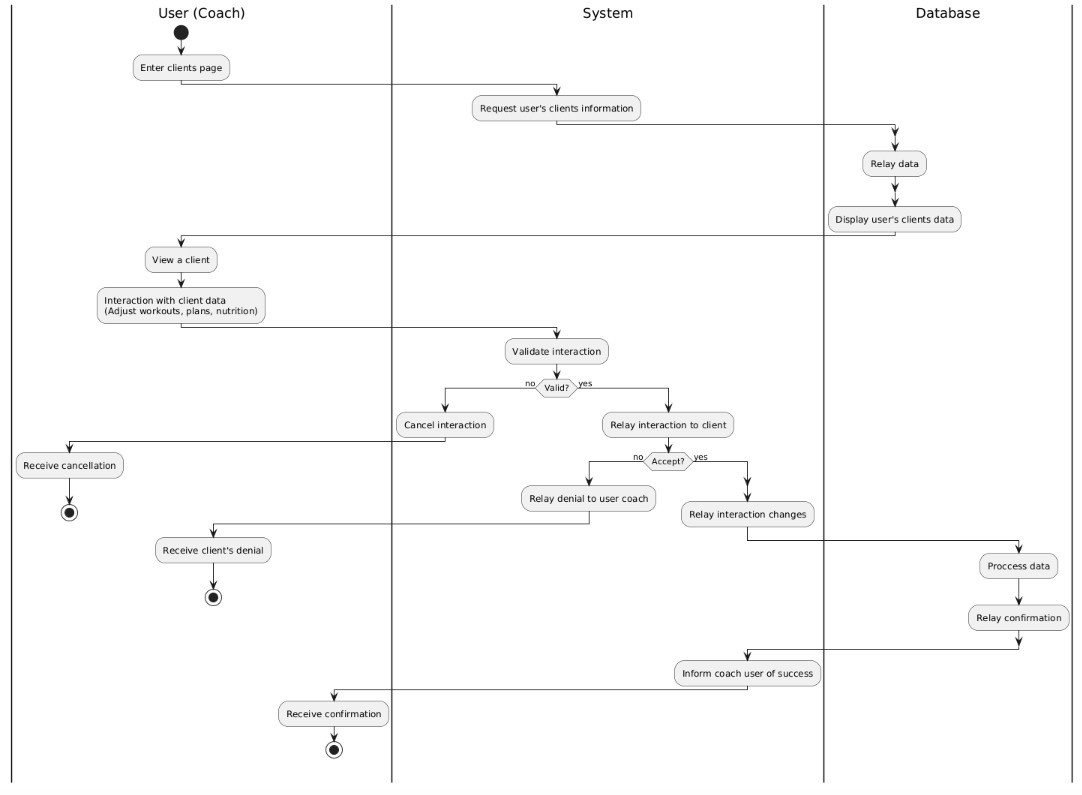
A diagram of a system

AI-generated content may be incorrect.

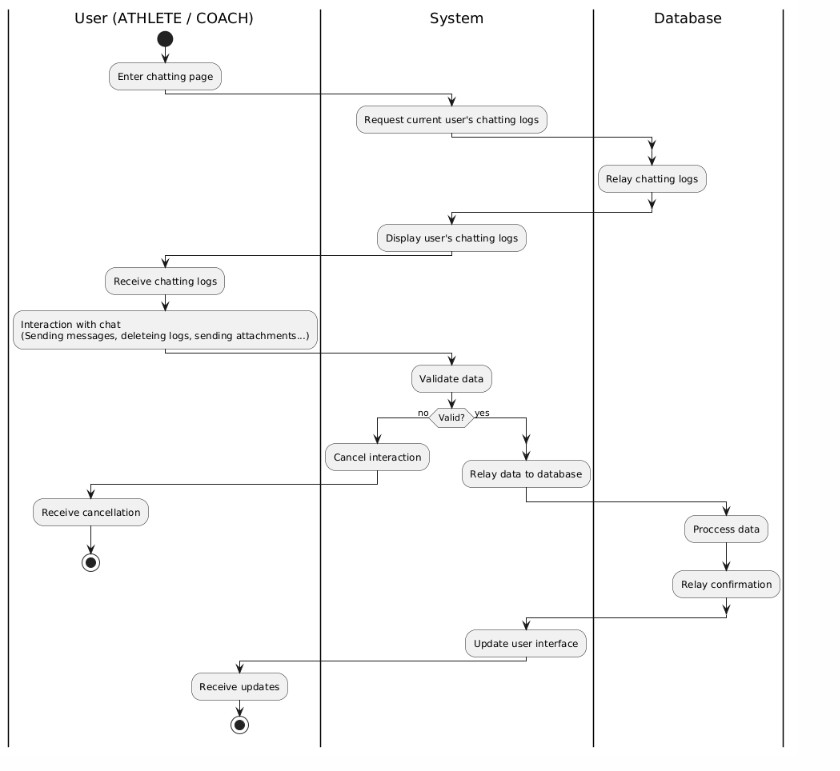
**Nutrition Workout Public Hub**

****

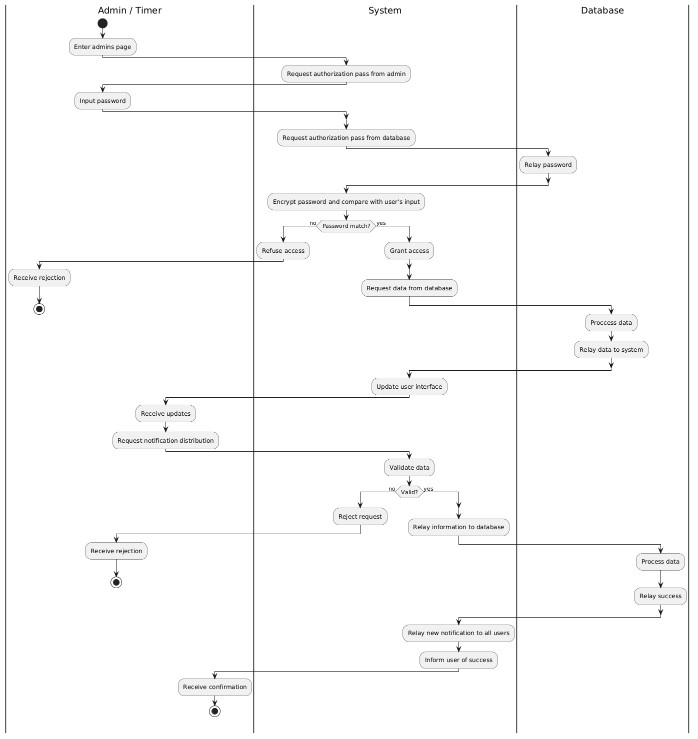
**Manage Client**



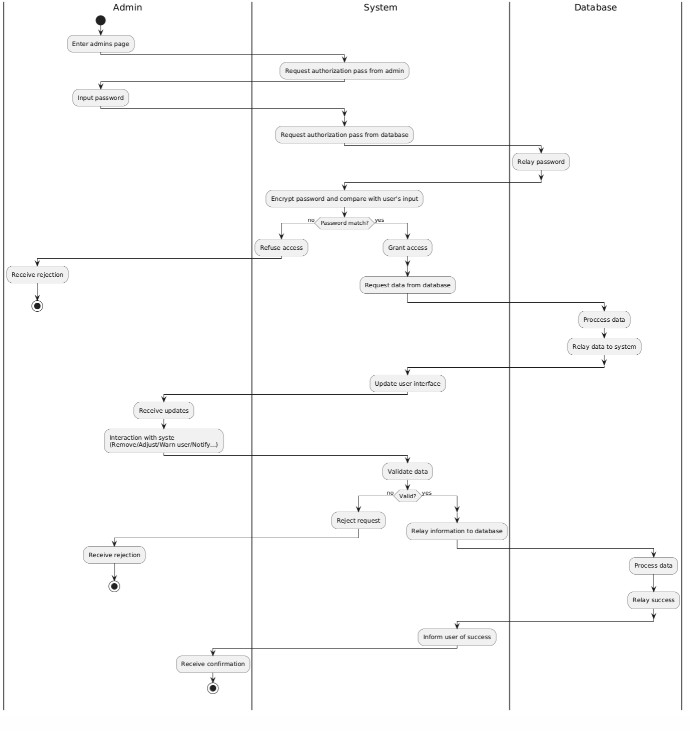
**Chatting**

****

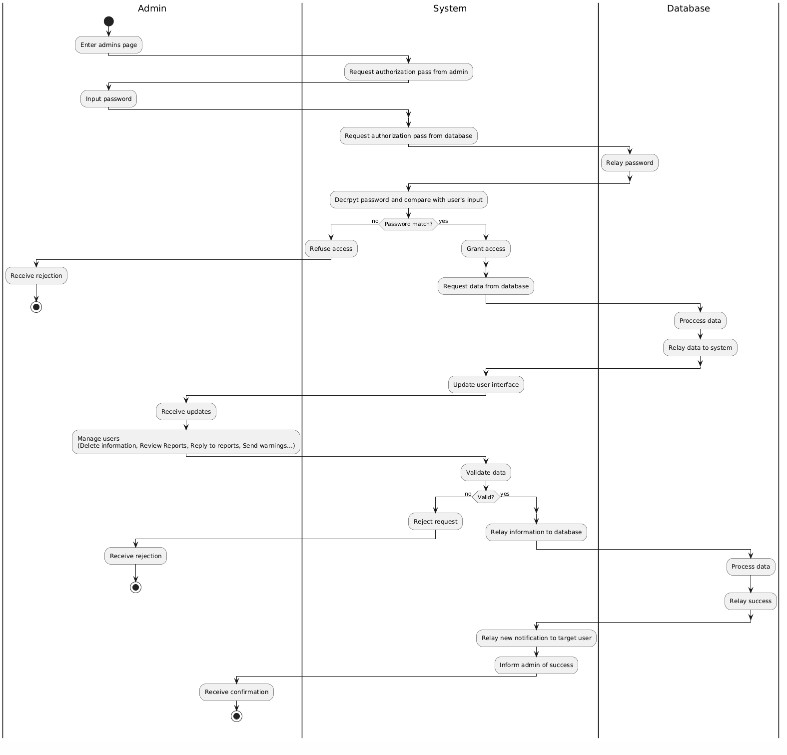
**Sending Notifications to Users**

****

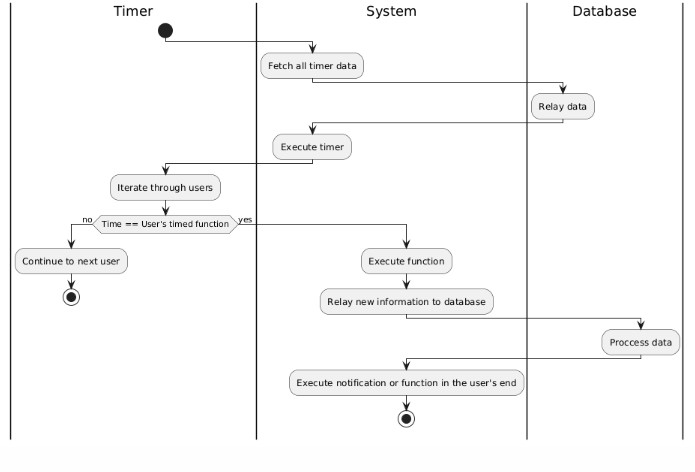
**Manage Public Hub**

****

**Manage Users**

****

**Timer**

****

1. **תיאור המרכיב האלגוריתמי - חישובי**

**10.1 הבעיה שבא לפתור**

The Kratos Hub is a sports app that aims to merge both nutrition and training into one application. The main points that the applications are meant to fix are the following:

* An app that allows athletes of different sports to interact in one whole community.
* Statistics for better understanding of progress.
* Overall, all-in-one app for all sports related logging.

**10.2 איסוף מידע וניתוחים סטטיסטיים**

* Analysis of most exercises utilized for each body part.
* Analysis of the most utilized muscle groups in different sports.

Ex. Leg muscle groups (Quads, Hamstrings and calves) are mostly exhausted in running marathons.

* Analysis of the best coaches and athletes in the industry to imitate and improve on their success.
* Analysis of the factors that go into an athlete’s success and progress.
* Analysis of what mostly brings individuals of the fitness industry together, what’s relevant and what’s not and how they relate to each other.
* Analysis of common causes of an athlete’s failure in their sport.
* Analysis of most used foods in an athlete’s diet, data that allows us to automatically suggest meals for the user.

1. **אבטחת מידע**

* **User Information:**

User will be encrypted one by one, and they will arrive in also encrypted packages. This ensures that they remain secure and hard to breach.

Upon registration, certain rules and legal string sequences will have to be followed when entering information to make sure their information is even more secure.

Passwords are hashed and irreversible which means when a user logs in, their input is hashed and stored hash compares to hashed input and return a response.

* **User Messages:**

All messages between users and exchanged information will always be encrypted during transit. Each side of both users will have key unique for their relationship that will be able to decrypt information between them and make them readable to them.

* **Tokens & Server Access:**

Certain CORS links will be guarded by a validation system that will only allow requests with certain tokens attached to have access to the server. This way, the user and only the user can access their information. The token will be stored in their device’s respective secure storage system where it will be encrypted by native device systems.

The token stored in the user’s device is a refresh token that is eligible for 6 months. Each time the user tries to open the app, the refresh token is used to create session token that will be used to access CORS

1. **משאבים הנדרשים לפרויקט**

**12.1 מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות**

At least 8 hours, 5 days of every week must be dedicated to the project.

Division of work will be based on each member’s skills and favorite field such as UI design, Backend development, Logical programming…

**12.2 ציוד נדרש**

A computer... duh and a board to draw and keep track of architectural flow.

Headphones and music to prevent losing my sanity.

* 1. **תוכנות נדרשות**
* Visual Studio Code for React Native & Node.js.
* SQL Management Studio for building our SQL database.
  1. **ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט**
* The study of training and nutrition fields.
* Logical programming.
* Regex programming.
* Database & data security.
* Design.
  1. **ספרות ומקורות מידע**
* Self-taught information (experience).
* www.wikipedia.com
* www.youtube.com
* Several websites are dedicated to coding.
  + www.stackoverflow.com
  + www.w3schools.com
  + www.codecademy.com
* Information collected from other applications.

1. **תכנית עבודה ושלבים למימוש הפרויקט**

* אפיון – ימשך עד 27/06/2024
* עיצוב המערכת וקידוד – ימשך עד 31/8/2025
* בדיקות ותיקונים – במהלך 9/2025
* הגשת הפרוייקט – עד לתאריך 1/10/2025
* הגנה על הפרוייקט - במהלך 10/202

1. **בדיקות**

**14.1 בדיקות תהליכיות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result** | **Expectation** | **Input / Scenario** | **Description** | **Test ID** |
| Passed | User creation | Fill form with valid data | User registration | PT-01 |
| Passed | Login successful | Enter valid email & password | User Login | PT-02 |
| Passed | Meal logged correctly | Log daily meal & nutrients | Nutrition Log | PT-03 |
| Passed | Metrics updated, affected metrics also recalculated | Update weight & height | Metrics Update | PT-04 |
| Passed | Message logged | Send message | Chat | PT-05 |
| Passed | Plan created | Assign plan to user | Workout Plan Assignment | PT-06 |
| Passed | Workout logged | Log performed workout | Workout Logging | PT-07 |
| Passed | User and related data removed | Delete user | Data Deletion | PT-08 |
| Passed | Email sent | Trigger on recovery | Email Verification | PT-09 |

**14.2 בדיקות יחידה**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Result** | **Expectation** | **Input** | **Test Description** | **Function** | **Test ID** |
| Passed | All are added to the database | User object, Metrics Object, Nutrition Object | Creating new user in the database | insertUser() | UT-01 |
| Passed | Columns in update object updated | ID, Token, Update object | Updating user, information can be multiple | updateUser() | UT-02 |
| Passed | User and related data permanently deleted | ID, Token | Deleting a user and related data | deleteUser() | UT-03 |
| Passed | Meal inserted into DB for use | ID, Token, Meal object | Logging new meal | logMeal() | UT-04 |
| Passed | Plan inserted and client receives notification | ID, Token, Client ID, Plan Object | Assigning plan to athlete client | assignPlan() | UT-05 |
| Passed | Workout inserted into DB | ID, Token, Workout Object | Logging new workout | logWorkout() | UT-06 |
| Passed | Password comes out hashed and stored in DB | Password | Hashing password in server | hashPassword() | UT-07 |
| Passed | Message logged in DB | ID, Token, Message | Sending messages through sever | sendMessage() | UT-08 |
| Passed | Content should reach the client’s email | ID, Token, Email, Mail Content | Sending an email via server | sendEmail() | UT-09 |

1. **בקרת גרסאות**

All steps are done with front-end, server and DB working in line together rather than building each tier separately.

|  |  |  |
| --- | --- | --- |
| **Test Description** | **Date** | **Version** |
| First instance of front-end created, basic layout and styling were being made. | 2025-04-23 | v0.1.0 |
| Registration, recovery and login completed with functional server coordination. | 2025-05-02 | v0.2.0 |
| Profile management and visual representation of user information completed. | 2025-06-12 | v0.3.0 |
| Goals and metrical functionality completed. | 2025-09-01 | v0.4.0 |
| Nutrition section completed. | 2025-09-23 | v0.5.0 |
| Final product is expected to release on that date. | 2025-09-15 | Version 1.0.0 |

1. **קטעי קוד**

For code snippets, we will showcase an example of users creating a new meal log.

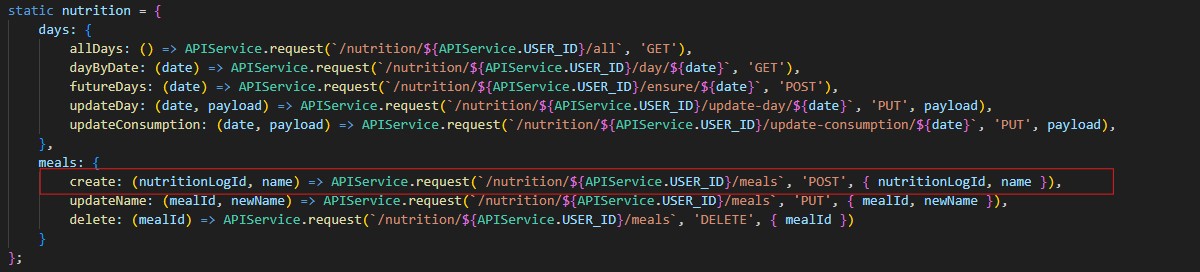
STAGE 1 - FRONT-END REQUEST

* This is a function triggered by a button in the UI. The function fires up a createInput react hook we created, one of many hooks responsible for creating a custom popup with each having its own job. createInput makes a popup that allows the user to enter an input and press confirmation.
* After the user enters a title, the title is extracted using array deconstruction (the values are stored as array to allow multiple inputs). Then a request is made to the backend (const result = await APIService….).
* A screen shot of a computer code

  AI-generated content may be incorrect.If successful, the response will contain the new meal object, it is then added to the list of user nutritionLogs and then app state will re-render. Else, the program will createToast (another one of our hooks) that will display the error message for the user.
* In here this is the APIService class mentioned earlier, it is responsible for forwarding calls to the backend server. The class contains routes, each responsible for different backend requests. The route then calls the request function and forwards parameters such as backend endpoint, CORS method, payload etc...

A screen shot of a computer program

AI-generated content may be incorrect.

* The route we called in handleMealAddition in the previous page

STAGE 2 - BACKEND-END STRUCTURE

* Before showing how the request is processed, we will first showcase the structure and environment of the server. This is index.js this is where the whole backend is fired up. It’s got one function, run(). This would fire up the Server, creates a singleton for it and it will handle all front-end requests.

A computer screen with colorful text

AI-generated content may be incorrect.

* A screen shot of a computer code

  AI-generated content may be incorrect.This is server class, it initializes critical parts of the server such as Databse class, EmailService and ServerRouter for handling routes.

STAGE 3 - REQUEST PROCESSING

* This is one of the routers the server router is responsible for, the NutritionRouter, it contains all routes for nutrition requests such as the one we made to create a new meal, the one with link /:id/meals of method PUT.
* This is where security measurements are done such as token validation. If the token is invalid, the request will return error 401.

A computer screen shot of a person

AI-generated content may be incorrect.

* The router sends the request to a controller for processing. This is NutritionController, it handles requests routed from the NutritionRouter. For our request, we want to create a meal for a user of a specific ID. After making sure all information is correct, the request is passed to the DatabaseService in this case NutritionDatabaseService.

A computer screen with text on it

AI-generated content may be incorrect.

* In the DatabaseService, the information is processed, then inserted. And this is where the result is sent back to the front-end.

A screen shot of a computer screen

AI-generated content may be incorrect.

EXTRA

For a more complicated and heavier DBService task, this is how user information is updated.

* In our app, information about the user is endless, there is user profile, metrics and nutrition. To make it a sustainable and maintainable process, an SQL transaction is first created. SQL transactions allow us to roll back any changes in the case of one of the following request functions fails. These functions are:
  + updatePassword - hashing, validating and storing new passwords.
  + updateProfile - updating new profile information.
  + updateMetrics - updating new metrical and bodily information.
  + updateNutrition – updating new information about nutrition.

A computer screen shot of text

AI-generated content may be incorrect.

* And this for example is the updatePassword function.

A screen shot of a computer code

AI-generated content may be incorrect.

END