Design

Document

**Gym**

**Management**

**System**

**Index**

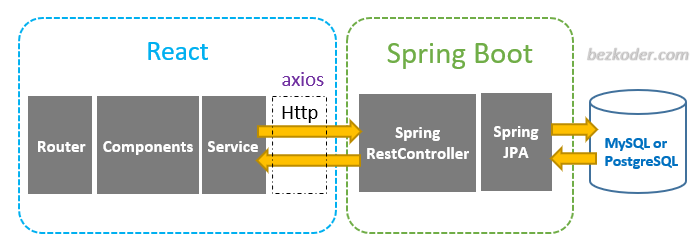
|  |  |  |
| --- | --- | --- |
| Sr No | Content | Page No |
| 1 | Introduction | 3 |
| 2 | Architectural Design | 3 |
| 3 | High Level Design | 6 |
|  | 3.1 E-R Diagram | 6 |
|  | 3.2 Page Navigation Diagram | 7 |
|  | 3.3 Data Flow Diagram | 9 |
|  | 3.4 Deployment Diagram | 12 |
| 4 | Low Level Design | 13 |
|  | 4.1 Database Design | 13 |
|  | 4.3 Details Of Page Navigation | 17 |

* **Introduction:**

This document is meant for the description of the structure and the database which we are using in this project. This document gives brief description about Architecture of the system, E-R diagram of the system and the table descriptions, the page navigation diagrams and the detail description for the page navigation.

* **Architecture Design:**

Following diagram shows the details of the Gym Management system architecture.



**First Tier (Client Tier - React):**

In the first tier, we focus on the user interface, which is the client tier. React, a powerful JavaScript library for building user interfaces, serves as our primary tool for creating an engaging and interactive front-end. React offers advantages such as a component-based architecture, efficient state management, and dynamic rendering, making it an ideal choice for this tier. AJAX is used to enhance interactivity and streamline navigation, ensuring a seamless user experience. JavaScript is employed for client-side validation, enhancing data integrity and providing real-time user feedback. HTML is used for structuring and presenting content, ensuring a user-friendly interface.

**Second Tier (Server Tier - Spring Boot):**

The second tier is divided into two primary components:

**Server API (Spring Boot with React):** This part of the second tier utilizes Spring Boot as a backend framework, serving as an API provider for the React-based frontend. React components communicate with the Spring Boot backend via RESTful APIs. Spring Boot is responsible for handling business logic, data processing, and serving JSON data to the React components. React Router is used for client-side routing to enable seamless navigation within the application.

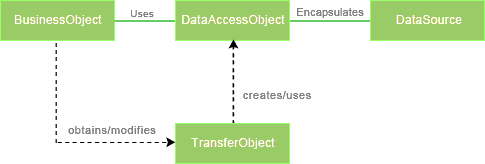
**Server Processing (Spring Boot with Java):** Spring Boot serves as the foundation for server-side processing, offering ease of development, platform independence, and extensive access to the Java ecosystem. It enables us to develop robust server-side applications that handle business logic efficiently. Spring Boot's servlet capabilities ensure smooth communication between the client (React) and the server (Spring Boot).

**Third Tier (Data Tier - MySQL and Spring Data JPA):**

The third tier consists of the Data Access Object (DAO) layer and the backend database using MySQL.

**Data Access Object (DAO - Spring Data JPA):** The DAO layer, implemented using Spring Data JPA, abstracts the interaction between the server and the database, adhering to best practices in software architecture. Spring Data JPA provides a clean and efficient way to perform database operations while abstracting the underlying data access implementation from the rest of the application. It ensures separation between the business logic layer and the persistent layer, promoting maintainability and scalability.

**Backend Database (MySQL):** MySQL serves as the backend database for the Vehicle Rental System. MySQL is a robust and widely-used relational database management system, offering data storage and retrieval capabilities. It stores and manages the application's data, ensuring data integrity, security, and reliability.

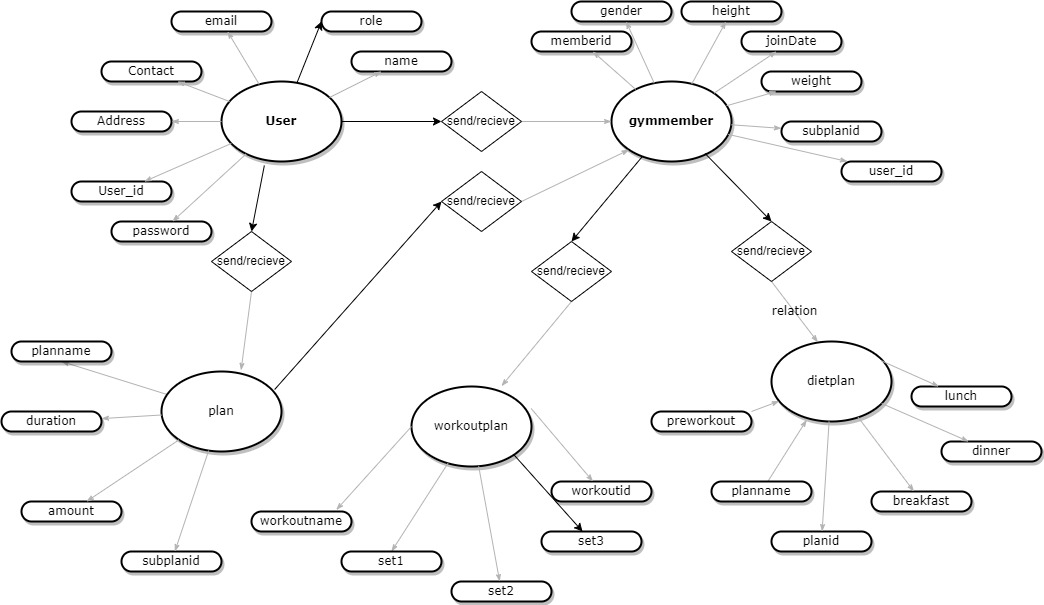


*Fig Data Access Object (DAO) Mechanism*

**Java Database Connectivity (JDBC):**

JDBC is used to provide database connectivity from java to database. Using Java database connectivity we can update/retrieve data to/from database with java programs. The main advantage of using JDBC is we can execute database queries by the program so that we can utilize the functionality provided by the database (with the queries). More over we can use triggers too. JDBC provides much other functionality (like the functions provided by CallableStatemtent class) to manage the data. Additionally, loading the driver will be different to different databases.

* **High Level Design:**
* **E-R Diagram:**

****

Above E-R Diagram shows that database of Gym Management System consist of following entities:

**1.`User**:

- `user\_id` (Primary Key): Unique identifier for each user.

- `address`: Address of user.

-`contact`: Contact of user.

- `Email` (Unique Key): Email address of the user.

- `name`: Name of the user.

- `password`: Password of user.

- `role`: role of user.

**2. `gymmember` Table:**

- `memberid` (Primary Key): Unique identifier for each member.

- `gender`: Gender of member.

- `height`: Height of member.

- `joindate`: Date of joining of member.

- `weight`: Weight of member.

- `subplan\_id` (Foreign Key): Identifier of the member associated with the membership plan of member.

- `user\_id`(Foreign Key): Identifier of the member associated with the user\_id of member.

**3. `plan` Table :**

- `subplan\_id` (Primary Key): Unique identifier for each plan.

- `amount`: Amount for membership of gym.

- `duration`: Duration for membership of gym.

- `planName`: Name od plan.

**4. `dietplan` Table:**

- `Planid` (Primary Key): Unique identifier for each dietplan.

- `breakfast `: Diet for breakfast.

- `dinner`: Diet for dinner.

- `lunch`: Diet for lunch.

- `planname`: Name of dietplan.

- `preworkout`:preworkout’s before main workout .

**5. `workoutplan` Table:**

- `workoutid` (Primary Key): Unique identifier for each workoutplan.

- `set1`: Set 1 of workout.

- `set2`: Set 2 of workout.

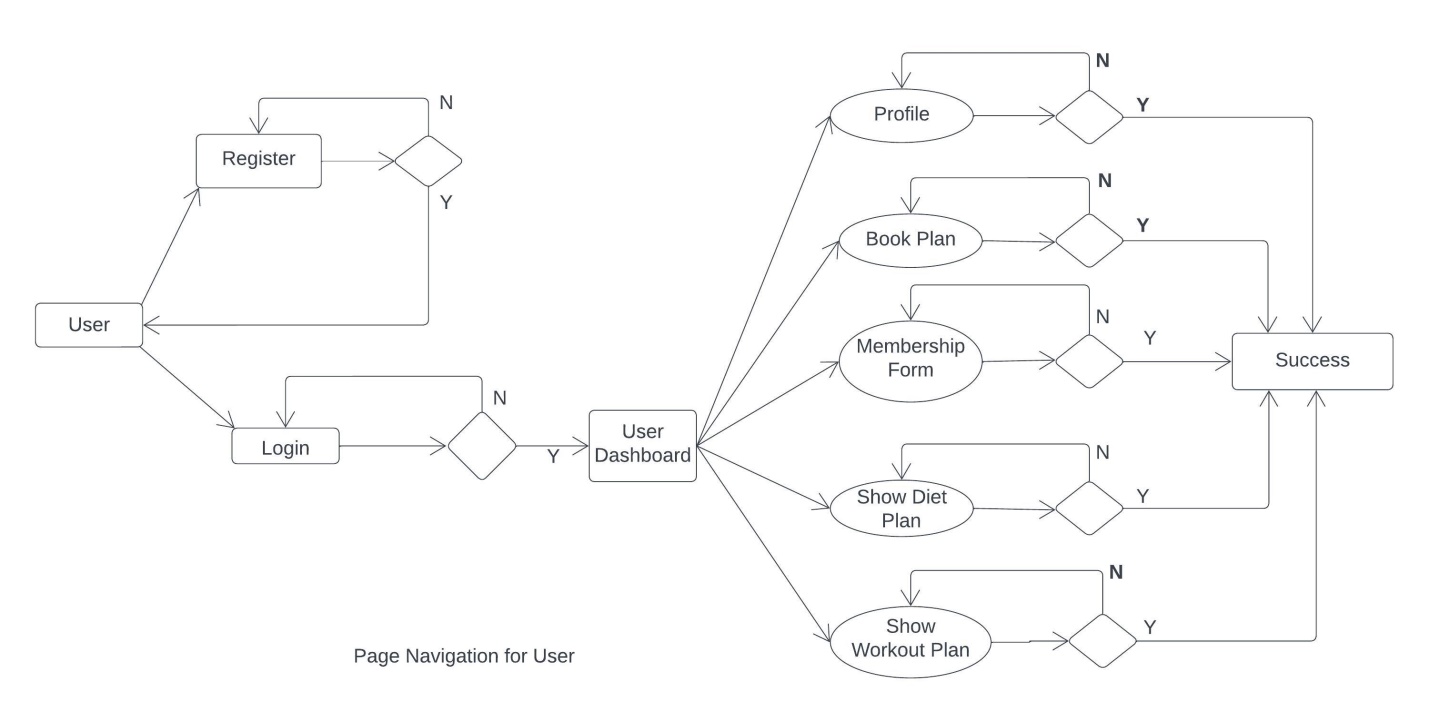
- `set3`: Set 3 of workout.

- `workoutname` : name of workoutplan.

**Page Navigation Diagram:**

* **User**

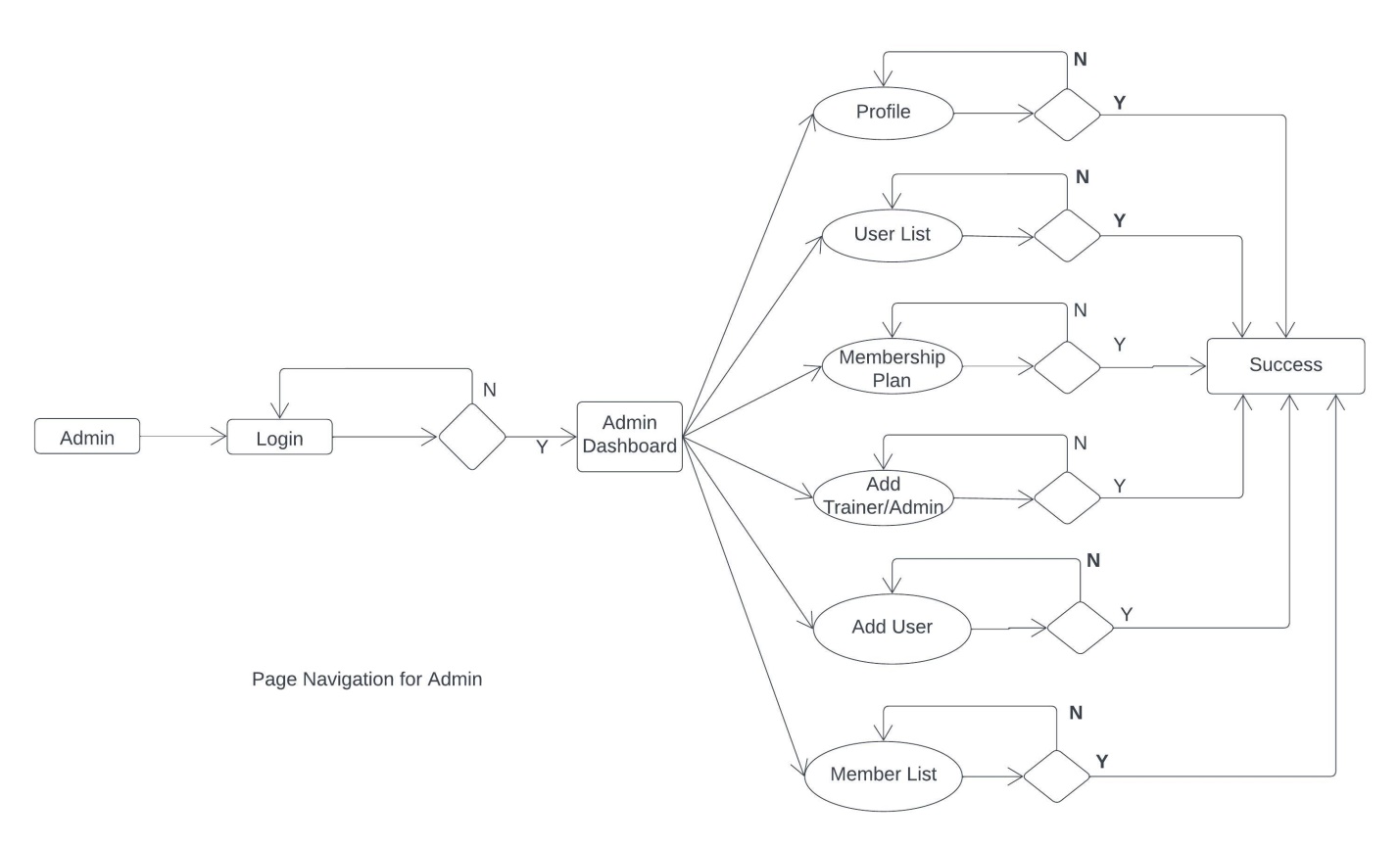
Following diagram explains the page navigation for the user module



**Fig.: Page Navigation For User**

* **Admin**

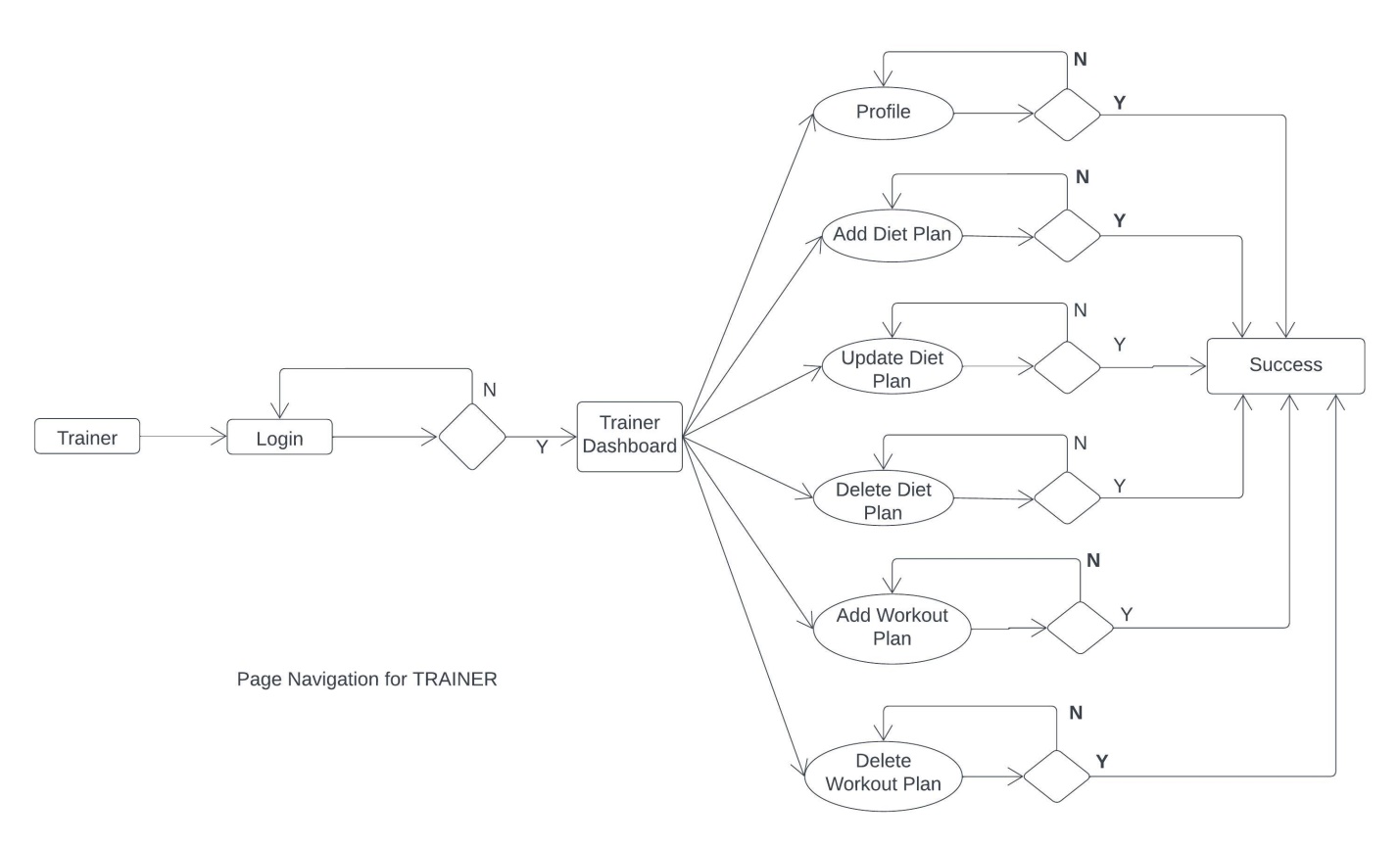
Following diagram explains the page navigation for the Admin module:



**Fig.: Page Navigation for admin**

* Trainer

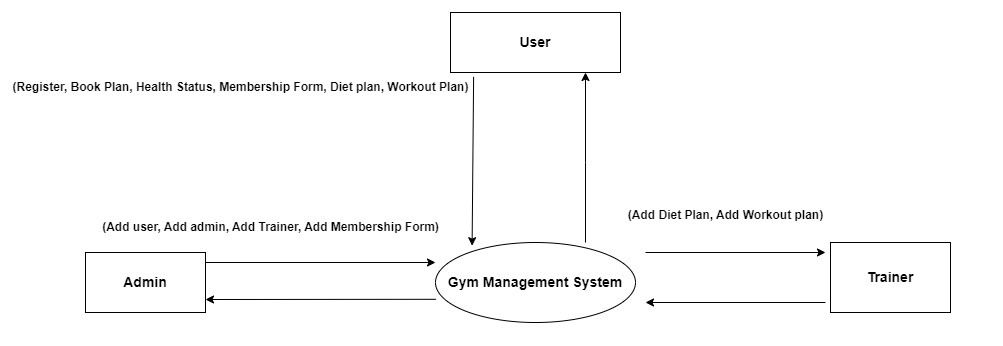
Following diagram explains the page navigation for the Trainer module:



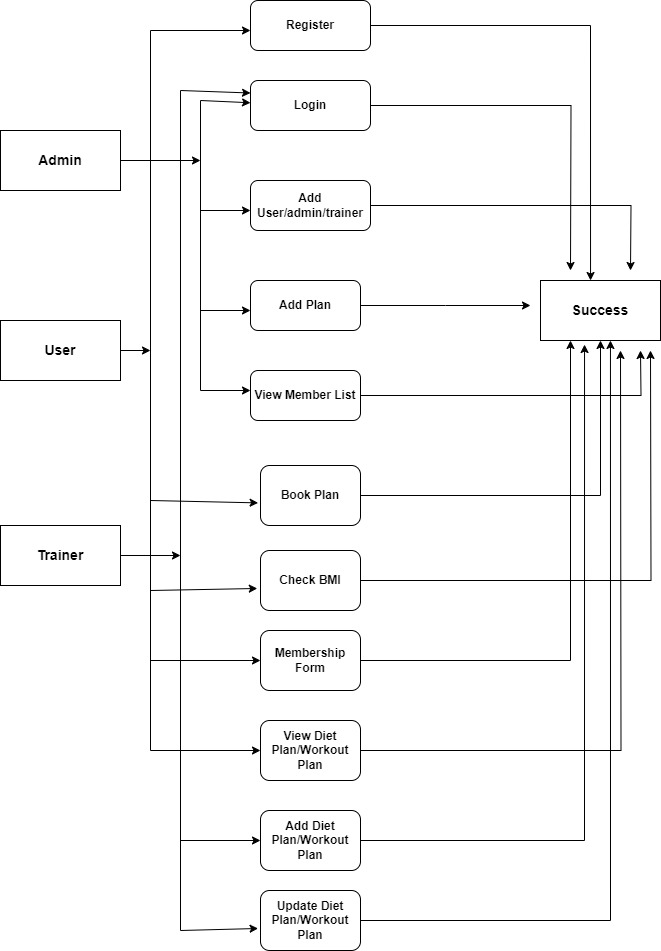
**Fig.: Page navigation for trainer**

**Data Flow Diagram:**

* **0-Level DFD:**

****

* In 0-Level DFD, there are three Entities:
* User
* Admin
* Trainer
* **1-Level DFD:**

****

*Fig. 1-Level DFD*

In 1-Level DFD, Admin Entity having following processes:

* Login
* Add user/admin/trainer
* Add plan
* View memberlist

User Entity having following processes:

* Register
* Login
* Book plan
* Check BMI
* Membership form
* View dietplan/workoutplan

Trainer Entity having following processes:

* Login
* Add dietplan/workoutplan
* Update dietplan/workoutplan

**Low Level Design:**

**Database Design:**

**1] user**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Description** |
| user\_id | Int | No | Primary key | Auto-increment | User ID |
| address | Varchar(255) | No |  | Null | Address of User |
| contact | Varchar(255) | No |  | Null | Contact of User |
| email | Varchar(255) | No |  | Null | Email of User |
| name | Varchar(255) | No |  | Null | Type of User |
| password | Varchar(255) | No |  | Null | Account Password |
| role | Varchar(255) | No |  | Null | Role (admin / trainer / user) |

**2] gymmember**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Description** |
| memberId | Int | No | Primary key | Auto-increment | Id of member |
| gender | Varchar(255) | No |  | Null | Gender of member |
| height | Double | No |  | Null | Height of member |
| joinDate | Date | No |  | Null | Date of joining at gym of member |
| weight | Double | No |  | Null | Weight of member |
| subplanid | Int | No | Foreign key | Null | Payment plan id |
| User\_id | Int | No | Foreign key | Null | User Id |

**3] plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Description** |
| subplanId | Int | No | Primary key | Auto-increment | Plan id for payment |
| amount | Int | No |  | Null | Amount as per plan |
| duration | Int | No |  | Null | Duration |
| planName | Varchar(255) | No |  | Null | Name of plan |

**4] dietplan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Description** |
| planid | Int | No | Primary key | Auto-increment | Dietplan Id |
| breakfast | Varchar(255) | No |  | Null | Diet for breakfast |
| dinner | Varchar(255) | No |  | Null | Diet for dinner |
| lunch | Varchar(255) | No |  | Null | Diet for lunch |
| planname | Varchar(255) |  |  | Null | Name of plan |
| preworkout | Varchar(255) |  |  | Null | Preworkout before workout |

**5] workoutplan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Description** |
| workoutid | Int | No | Primary key | Auto-increment | Id of workout plan |
| set1 | Varchar(255) | No |  | Null | Set 1 for workout |
| set2 | Varchar(255) | No |  | Null | Set 2 for workout |
| set3 | Varchar(255) | No |  | Null | Set 3 for workout |
| workoutname | Varchar(255) | No |  | Null | Name of workoutplan |