**WEIGHT FOOD TRACKER**

**INTRODUCTION:**

This application tracks the food intake of different people of different age and weight category according to their diet plan. It tracks whether they are following regular and healthy food habits with proper diet plans and maintaining good health. It records the details of all the Consumers.

This application uses REST API to perform CRUD Operations with the help of SQL Server for the Consumers and unit testing with Nunit and End to End testing for the testing function.

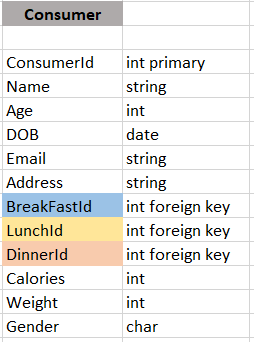
**DATABASE:**

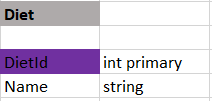
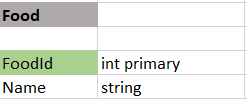
1. **Entities:**

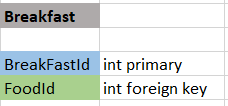
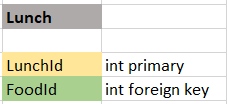
Entity is defined as a thing, person, place, unit, object or any item about which the data should be captured and stored. Following are the entities available in this API application:

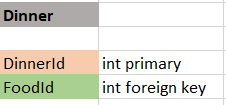
* Consumer ID
* Consumer Name
* DOB
* Age
* Gender
* Weight
* Address
* Mobile
* Email
* Diet plan
* Food
* Breakfast
* Lunch
* Dinner
* Calories
* Address

1. **Table Design:**







1. **Primary and Foreign Keys:**

**Consumer Table:**

Primary Key - ConsumerId

Foreign Keys – DietId, BreakFastId, LunchId, DinnerId

**Diet Table:**

Primary Key - DietId

**Food Table:**

Primary Key - FoodId

**Breakfast Table:**

Primary Key - BreakFastId

Foreign Keys – FoodId

**Lunch Table:**

Primary Key - LunchId

Foreign Keys – FoodId

**Dinner Table:**

Primary Key - DinnerId

Foreign Keys – FoodId

**4.SQL Queries:**

CREATE DATABASE FoodWeightDB

Use FoodWeightDB

CREATE TABLE Diet ( DietId int primary key, Name varchar(50))

CREATE TABLE Food (FoodId int primary key, Name varchar(50))

CREATE TABLE Breakfast (BreakFastId int primary key, FoodId int )

ALTER TABLE Breakfast ADD CONSTRAINT FK\_Breakfast\_Food FOREIGN KEY (FoodId) REFERENCES Food (FoodId);

CREATE TABLE Lunch ( LunchId int primary key, FoodId int )

ALTER TABLE Lunch ADD CONSTRAINT FK\_Lunch\_Food FOREIGN KEY (FoodId) REFERENCES Food(FoodId);

CREATE TABLE Dinner (DinnerId int primary key, FoodId int )

ALTER TABLE Dinner ADD CONSTRAINT FK\_Dinner\_Food FOREIGN KEY (FoodId) REFERENCES Food (FoodId);

CREATE TABLE Consumer(ConsumerId int primary key, Name varchar(50), Age int, DOB date, Gender char, Weight int, Email varchar(50),Address varchar(150),DietId int, BreakFastId int, LunchId int, DinnerId int, Calories int)

ALTER TABLE Consumer ADD CONSTRAINT FK\_Consumer\_Breakfast FOREIGN KEY (BreakFastId) REFERENCES Breakfast (BreakFastId);

ALTER TABLE Consumer ADD CONSTRAINT FK\_Consumer\_Lunch FOREIGN KEY (LunchId) REFERENCES Lunch (LunchId);

ALTER TABLE Consumer ADD CONSTRAINT FK\_Consumer\_Dinner FOREIGN KEY (DinnerId) REFERENCES Dinner (DinnerId);

ALTER TABLE Consumer ADD CONSTRAINT FK\_Consumer\_Diet FOREIGN KEY (DietId) REFERENCES Diet (DietId);

**API:**

**Authentication with Login Method:**

This is to validate the Consumer who is logging in to perform functions using the API. This authentication is done with the help of jwt tokens.

Method Name: Login

Description: Login with proper credentials

Parameters: 1. **Username**-admin **password** – admin123

2. **Username**-Consumer **password** –user123

Returns: role (admin/Consumer) if Username and password are valid.

Error message if Username and password are invalid.

**CRUD API:**

CRUD is an acronym for the four basic SQL commands: Create, Read, Update and Delete. Most applications have some kind of CRUD functionality. A CRUD API is one that uses API to get data into and out of a database. Following are the operations used in this project:

**Read:**

Read function is used to retrieve the data already present in the database. It uses the SELECT query.

***Pseudo code***:

1. Method Name: GetConsumer

Description: Get required details of a Consumer based on Consumer Id

Parameter: Consumer Id

Returns: Required details of the Consumer

DB Table: Consumer

1. Method Name: GetAllConsumers

Description: Get required details of all Consumers

Parameter: None

Returns: Required details of all Consumers

DB Table: Consumer

**Create:**

Create function helps to insert a data into a table present in a database. It uses INSERT Query.

***Pseudo code***:

Method Name: InsertConsumer

Description: Create a new Consumer

Parameter: None

Returns: ConsumerId

DB Table: Consumer

**Update:**

Update function helps to make changes(update) to some data that is already inserted into a table. It uses UPDATE Query.

***Pseudo code***:

Method Name: UpdateConsumer

Description: Update details of a Consumer based on Consumer Id

Parameter: ConsumerId

Returns: Number Of Rows

DB Table: Consumer

**Delete:**

Delete function is used to remove data from the table, i.e Delete data. It uses DELETE Query.

***Pseudo code***:

Method Name: DeleteConsumer

Description: Delete details of a Consumer based on Consumer Id

Parameter: ConsumerId

Returns: ConsumerId

DB Table: Consumer

Method Name: DeleteAllConsumers

Description: Delete all Consumers

Parameter: None

Returns: Count of Consumers Deleted

DB Table: Consumer

**Testing:**

1. **Test Plan:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Step Number | Description | Return Value | Expected | Actual |
|  |  |  |  |  |  |
| CT- 001 | 1 | Insert Consumer | Consumer Id | >0 | >0 |
|  | 2 | Get Consumer for inserted ConsumerId | Inserted Consumer | NOT NULL Name = Inserted name | NOT NULL Name = Inserted name |
|  | 3 | Update Consumer | Number of Rows | = 1 | = 1 |
|  | 4 | Get Consumer for updated Consumer | Updated Consumer | NOT NULL Name = Updated name | NOT NULL Name = Updated name |
|  | 5 | Delete Consumer | Has Been Deleted | True | True |
|  | 6 | Get Deleted Consumer | Consumer | NULL | NULL |

1. **Unit Testing:**

Unit Testing is a software testing where individual units of an application are tested. The purpose is to validate that each unit of the application are designed perfectly. This project uses Nunit testing to test the API and for unit testing.

These are the tests that are performed in this API application:

**Login Test:**

This testing is to check whether the login function works or not when the credentials are given correctly.

**Login Fail Test:**

This testing is to check whether the login function returns error message or not when the credentials wrong.

**Get Consumers Test:**

This tests whether all the data stored is retrieved from the database.

***Pseudo code***:

Method Name: GetConsumers

Description: Get required details of a Consumer based on Consumer Id

Parameter: None

Returns: Required details of the Consumers

DB Table: Consumer

**Get Consumer Test:**

This tests whether the data stored for a particular id is retrieved from the database.

***Pseudo code***:

Method Name: GetConsumer

Description: Get required details of a Consumer based on Consumer Id

Parameter: Consumer Id

Returns: Required details of the Consumer

DB Table: Consumer

**Insert Consumer Test:**

This tests whether the given data is stored in the database, then retrieves the inserted data by getconsumer function.

After that it test to updates the given data to the retrieved data and tests to delete the updated data.

1. **API Testing:**

API Testing helps to test whether the CRUD Operations are performed in the API with the help of Nunit testing method ,http access and threading functions.

Following are the API Testing methods:

**CreateConsumerAsync :**

This method tests to insert the statically given data into the table and returns the inserted data in the form of json format with the help of http client.

***Pseudo code:***

Method Name: CreateConsumerAsync

Description: Insert given details of a Consumer into the database with API using http request.

Parameter: None

Returns: Inserted details of the Consumer in json format

DB Table: Consumer

**GetConsumerAsync :**

This method tests to retrieve already inserted data from the table and represents in json format with the help of http client request.

***Pseudo code:***

Method Name: GetConsumerAsync

Description: Get required details of a Consumer based on Consumer Id with API using http request

Parameter: Consumer Id

Returns: Required details of the Consumer in json format

DB Table: Consumer

**GetConsumersAsync :**

This method tests to retrieve all the inserted data from the table and represents in json format with the help of http client request.

***Pseudo code:***

Method Name: GetConsumersAsync

Description: Get required details of a Consumer with API using http request

Parameter: None

Returns: Required details of the Consumer in json format

DB Table: Consumer

**UpdateConsumerAsync :**

This method tests to update or change already inserted data in the table and represents in json format with the help of http client request.

***Pseudo code:***

Method Name: UpdateConsumerAsync

Description: Update the details of a Consumer based on Consumer Id with API using http request

Parameter: Consumer Id

Returns: Details of the updated Consumer in json format

DB Table: Consumer

**DeleteConsumerAsync :**

This method tests to delete data from the table with the help of http client request.

***Pseudo code:***

Method Name: DeleteConsumerAsync

Description: Delete details of a Consumer based on Consumer Id with API using http request

Parameter: Consumer Id

Returns: Number of rows deleted

DB Table: Consumer