**API Documentation:**

Website Swagger Link : - <https://food-api-u9af.onrender.com/api-docs>

PostMan Link :- [https://food-api-u9af.onrender.com/api/](https://food-api-u9af.onrender.com/api-docs)deliveryprice

Below is the documentation for the foodCalucation endpoint, detailing its functionality, request/response formats, and error handling:

Endpoint:

* URL: /api/deliveryprice
* Method: POST
* Description: Calculate the total price of food delivery based on the provided parameters.
* Tags: Food

Request Format:

Body:

zone: string (required) - The delivery zone.

organization\_id: string (required) - The organization ID.

total\_distance: number (required) - The total distance of delivery in kilometers.

item\_type: string (required) - The type of item (perishable/non-perishable).  
 Response Format:

Success Response (200):

Content Type: application/json

Body:

{

"total\_price": number

}

Description: The calculated total price of food delivery.

Error Responses:

400 Bad Request:

Content Type: application/json

Body:

{

"message": string

}

Description: Indicates that there was a problem with the request. The message field provides information about the error.

404 Not Found:

Content Type: application/json

Body:

{

"message": string

}

Description: Indicates that no items were found matching the specified criteria.

Error Handling:

* If any of the required fields (zone, organization\_id, total\_distance, item\_type) are missing or invalid, a 400 Bad Request response is returned with an appropriate error message.
* If the total\_distance is not between 1 and 30 kilometers, a 400 Bad Request response is returned with an appropriate error message.
* If the item\_type is not 'perishable' or 'non-perishable', a 400 Bad Request response is returned with an appropriate error message.
* If the organization\_id is not an integer, a 400 Bad Request response is returned with an appropriate error message.
* If the zone or item\_type is not a non-empty string, a 400 Bad Request response is returned with an appropriate error message.
* If no items are found matching the specified criteria, a 404 Not Found response is returned with an appropriate error message.

Database Setup :

* Open PgAdmin :- Create a Database and tables
* Open PgAdmin portal :- right click on databases --> click database -> enter database name. DB has created
* Open query tool, and select the database want to work with
* Create a table

Here is sample creation of database  
  
CREATE TABLE organization (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL);  
  
CREATE TABLE item (

id SERIAL PRIMARY KEY,

type VARCHAR(255) CHECK (type IN ('perishable', 'non-perishable')),

description TEXT

);  
  
CREATE TABLE Pricing (

organization\_id INT REFERENCES Organization(id) ON DELETE CASCADE,

item\_id INT REFERENCES Item(id) ON DELETE CASCADE,

zone VARCHAR(255) NOT NULL,

base\_distance\_in\_km INT NOT NULL CHECK (base\_distance\_in\_km > 0),

km\_price NUMERIC(10, 2) NOT NULL CHECK (km\_price >= 0),

fix\_price NUMERIC(10, 2) NOT NULL CHECK (fix\_price >= 0),

PRIMARY KEY (organization\_id, item\_id, zone)

);

INSERT INTO organization (name) VALUES

('Organization 001'),

('Organization 002'),

('Organization 003'),

('Organization 004'),

('Organization 005'),

('Organization 006'),

('Organization 007'),

('Organization 008'),

('Organization 009'),

('Organization 010'),

('Organization 011'),

('Organization 012'),

('Organization 013'),

('Organization 014'),

('Organization 015'),

('Organization 016'),

('Organization 017'),

('Organization 018'),

('Organization 019'),

('Organization 020');

INSERT INTO item (type, description) VALUES

('perishable', 'Perishable item 1'),

('non-perishable', 'Non-perishable item 1'),

('perishable', 'Perishable item 2'),

('non-perishable', 'Non-perishable item 2'),

('perishable', 'Perishable item 3'),

('non-perishable', 'Non-perishable item 3'),

('perishable', 'Perishable item 4'),

('non-perishable', 'Non-perishable item 4'),

('perishable', 'Perishable item 5'),

('non-perishable', 'Non-perishable item 5'),

('perishable', 'Perishable item 6'),

('non-perishable', 'Non-perishable item 6'),

('perishable', 'Perishable item 7'),

('non-perishable', 'Non-perishable item 7'),

('perishable', 'Perishable item 8'),

('non-perishable', 'Non-perishable item 8'),

('perishable', 'Perishable item 9'),

('non-perishable', 'Non-perishable item 9'),

('perishable', 'Perishable item 10'),

('non-perishable', 'Non-perishable item 10');

INSERT INTO pricing (organization\_id, item\_id, zone, base\_distance\_in\_km, km\_price, fix\_price) VALUES

(1, 1, 'Zone A', 5, 1.5, 10),

(1, 1, 'Zone B', 5, 1.5, 10),

(1, 2, 'Zone A', 5, 1, 10),

(1, 2, 'Zone B', 5, 1, 10),

(2, 1, 'Zone A', 5, 1.5, 10),

(2, 1, 'Zone B', 5, 1.5, 10),

(2, 2, 'Zone A', 5, 1, 10),

(2, 2, 'Zone B', 5, 1, 10),

(3, 1, 'Zone A', 5, 1.5, 10),

(3, 1, 'Zone B', 5, 1.5, 10),

(3, 2, 'Zone A', 5, 1, 10),

(3, 2, 'Zone B', 5, 1, 10),

(4, 1, 'Zone A', 5, 1.5, 10),

(4, 1, 'Zone B', 5, 1.5, 10),

(4, 2, 'Zone A', 5, 1, 10),

(4, 2, 'Zone B', 5, 1, 10),

(5, 1, 'Zone A', 5, 1.5, 10),

(5, 1, 'Zone B', 5, 1.5, 10),

(5, 2, 'Zone A', 5, 1, 10),

(5, 2, 'Zone B', 5, 1, 10),

(1, 1, 'Central', 5, 1.5, 10),

(1, 2, 'Central', 5, 1, 10);

Once the database has been created, download the project.

Install the dependencies

* npm install
* Go to .env file and configure your database credentials, such as   
  PORT = your\_port

user = postgres

host = localhost

database = db\_name

password = password

db\_port = 5432 // this is default port of postgres

* Once .env file is configured, npm start to start server,
* You can access the Swagger documentation by clicking on the provided link. Once there, navigate to the POST route you want to test. Click on 'Try it out', then input the desired JSON data for the POST request. After that, click on 'Execute' to send the request and receive the response.
* To test the backend, ‘http://localhost:8000/api/deliveryprice’ copy this and past it in Postman/thunderclient or any api test website.
* Or go to browser and paste -> <http://localhost:8000/api-docs/> here you can test a swagger ui . Click on post -> try it out -> set the variable in request body -> scroll down click on exccute.
* To create a request in Postman for a POST route:
* Open Postman.

Create a new request by clicking on the "New" button in the upper left corner and selecting "Request."

In the request builder, enter the URL of your POST route.

Choose the HTTP method as "POST" from the dropdown menu next to the URL.

Click on the "Body" tab below the URL bar.

Select "raw" from the options below the body tab.

Choose JSON (application/json) as the type of data you want to send.

Enter your JSON request body in the text area.

Click on the "Send" button to execute the request and see the response.

Website Link : - <https://food-api-u9af.onrender.com/api-docs>