# Task Guide

Task 1: Product Management

- Create Product Schema ** Attributes
  - `ProductID`
  - `ProductName`
  - `Price`
  - `Description`

- Insert a Product
  - Method: POST
  - URL: http://localhost:4000/createproduct
  - Body (JSON):

```
{
  "ProductId": 1,
  "ProductName": "Tata Salt",
  "Price": 20,
  "Description": "Tata Salt Iodised Crystal Salt is sourced from the sea, is solar evaporated and hygienically packed"
}
```

  - Expected Response: "Product Add Successfully"

- Retrieve All Products
  - Method: GET
  - URL: http://localhost:4000/products
  - Expected Response: List of all products output

- Retrieve a Particular Product
  - Method: GET
  - URL: http://localhost:4000/products/:id
  - Header: Content-Type: application/json
  - Body (JSON):

```
{
  "ProductId": 1
}
```

  - Expected Response: Product details for ID 1

- Update a Particular Product
  - Method: PATCH
  - URL: http://localhost:4000/products/1
  - Body (JSON):

    {

    "ProductId": 3,

    "ProductName": "Ashirvada Salt",

    "Price": 55,

    "Description": "Iodised Crystal Salt is sourced from the sea, is solar evaporated and hygienically packed"

    }

  - Expected Response: Success message
- Delete a Particular Product
  - Method: DELETE
  - URL: http://localhost:4000/products/3
  - Expected Response: Success message

Task 2: User Management

- Create User Schema ** Attributes:
  - name
  - email
  - password
  - phonenumber
- Create a User
  - Method: POST
  - URL: http://localhost:4000/signup
  - Body (JSON):

    {

    "name": "Pandi",

    "email": "peswaran40@yopmail.com",

    "password": "User@123",

    "phonenumber": 1234567890

    }

  - Expected Response: "User Add Successfully"
- Login User
  - Method: POST
  - URL: http://localhost:4000/login
  - Body (JSON):

    {

    "email": "peswaran40@yopmail.com",

"password": "User@123"

}

- Expected Response: "Login Successfully" with token

- Access Protected Pages
  - Method: POST
  - URL: http://localhost:4000/profile
  - Token required in headers

Task 3: Data Fetching and In-Memory Cache

- Fetch Data from External API
  - Method: GET
  - URL: http://localhost:4000/external

- View Cache Memory
  - Method: GET
  - URL: http://localhost:4000/show-cache

Task 4: User Management

- Create a User
  - Method: POST
  - URL: http://localhost:4000/signupuser
  - Body (JSON):

{

"name": "Pandi",

"email": "peswaran40@yopmail.com",

"password": "User@123",

"phonenumber": 1234567890

}

- Expected Response: "User Add Successfully"

- Get All Users
  - Method: GET
  - URL: http://localhost:4000/alluser
  - Expected Response: List of all users
- Get Particular User by ID
  - Method: GET
  - URL: http://localhost:4000/user/:id
  - Expected Response: User details
- Update Particular User
  - Method: PATCH
  - URL: http://localhost:4000/user/:id
  - Body (JSON):

{

"name": "Eswaran",

"email": "peswaran40@yopmail.com",

"phonenumber": 0987654321

}

- Expected Response: Success message

- Delete Particular User
  - Method: DELETE
  - URL: http://localhost:4000/userdelete/:id
  - Expected Response: Success message

Task 5: Real-Time Update

- Set up the Server and Test Real-Time Updates
- Run the server: node server
- Open two browsers and navigate to http://localhost:8000/
- Verify: Real-time updates with side-by-side browser windows

Task 6: Error Handling

- Get Particular User by ID
- Method: GET
- URL: http://localhost:4000/users/:id
- Error Handling: If user is not found, log error
- Log File: Assignment -> logs -> error.log

Log Entry:

{

"level": "error",

"message": "Operational error: User not found"

}