FullStack Assignment

Guidelines:

i. ***This assignment is mandatory for everyone***

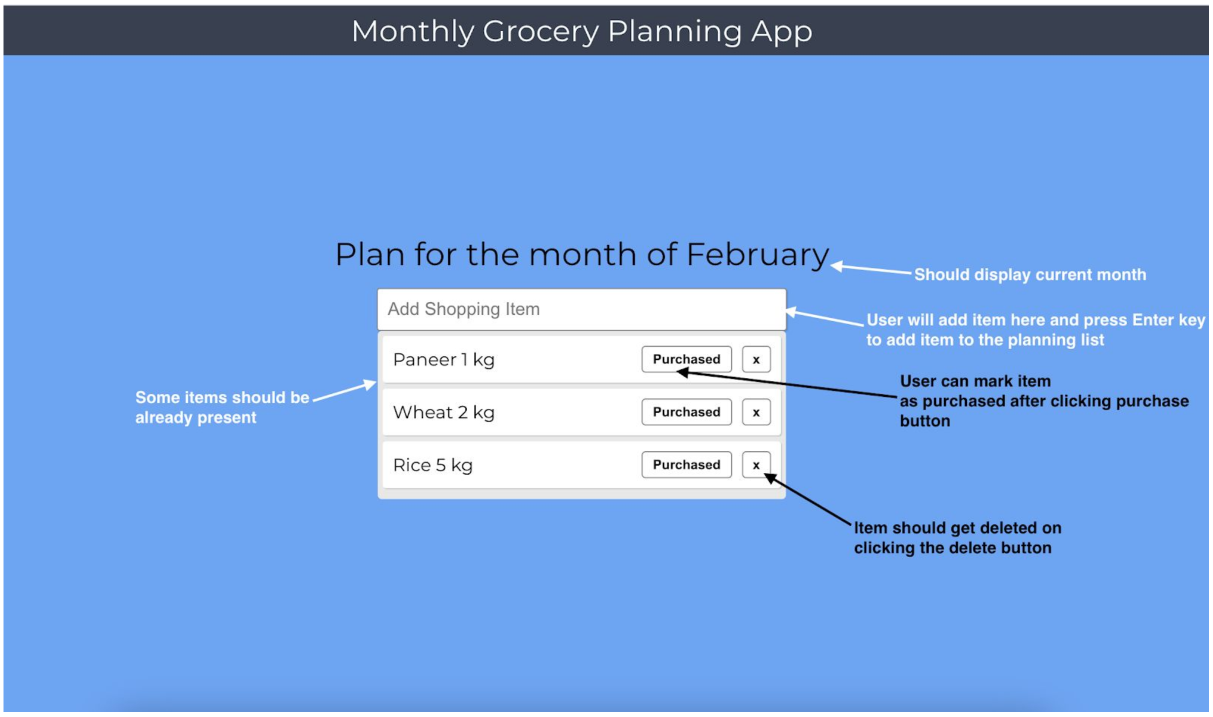
ii. *There will only be a single attempt for each exam and* ***no deadline extension in case of assignments***

iii. *Any case of unfair means or* ***plagiarism would lead to debarring*** *in final placements without any further consideration.*

iv. The assignment solutions should be uploaded on Github and links to Github repository links should be shared with the coach for code review. Make sure to add appropriate comments in code wherever possible.

Problem Statement:

Build a monthly Grocery Planning Application in which user should be able to add grocery items for monthly purchase, mark the items as purchased and delete the items from list as shown in below screenshot:



Should display current month

User will add item here and press Enter key to add item to the planning list

User can mark item as purchased after clicking purchase button

Some items should be already present

Item should get deleted on clicking the delete button

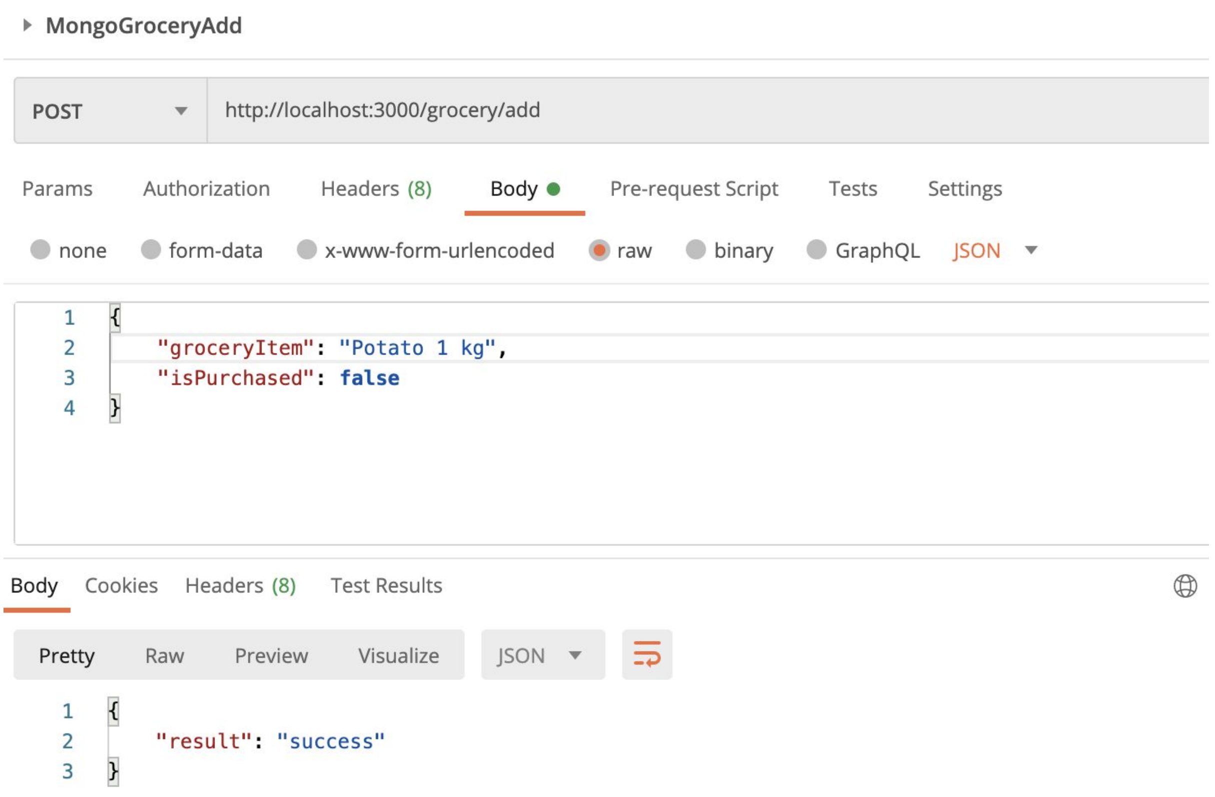
Steps:

Backend API development

Step 1:

Item Addition API:-

1. Create a connection with a local or remote Mongo database in express.js server
2. Build a model in express.js for grocery item details using mongoose
   The model should have following parameters:-
   groceryItem :- Name of grocery item
   isPurchased :- Track purchase status of a particular item
3. Next build a route to add a grocery item as shown below



POST http://localhost:3000/grocery/add

```
{
    "groceryItem": "Potato 1 kg",
    "isPurchased": false
}
```

Response Body (Pretty, JSON):

```
{
    "result": "success"
}
```

Request URL:- *http://localhost:3000/grocery/add*

Request Method:- POST

Sample Request JSON:-

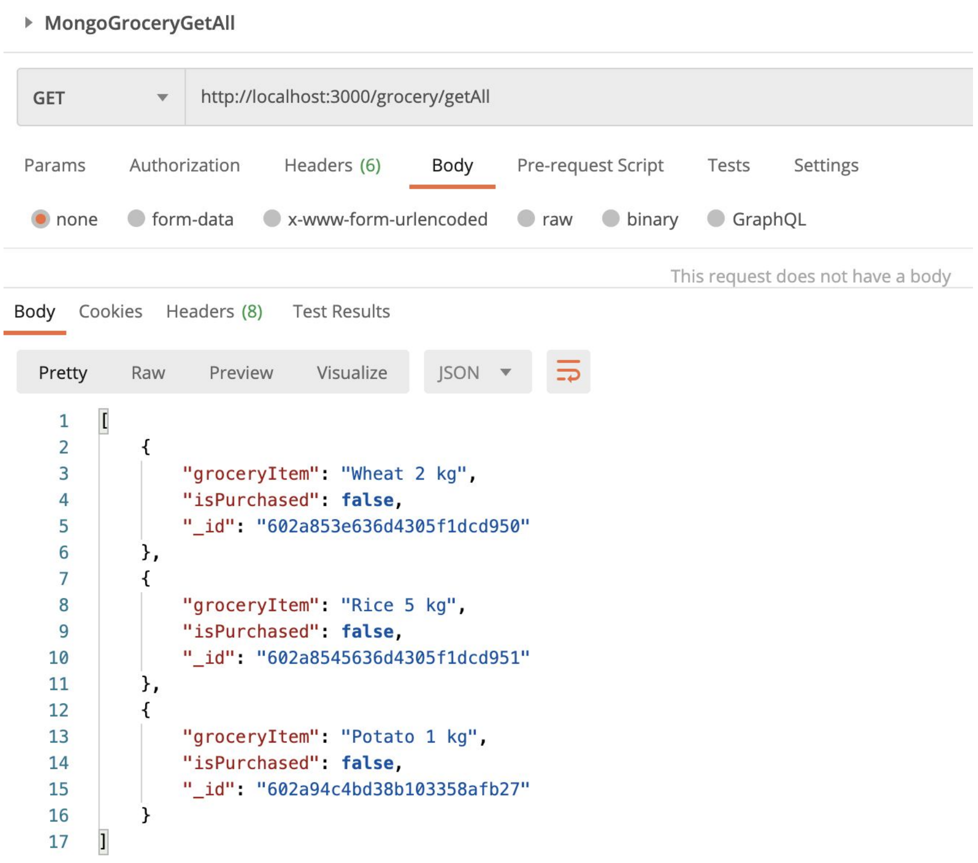
```
{
   "groceryItem": "Potato 1 kg",
   "isPurchased": false
}
```

Response JSON:-

```
{
    "result": "Success"
}
```

Step 2:

Once grocery details are saved to database, build another API which can be used to get all grocery items details as shown below:-



http://localhost:3000/grocery/getAll

This request does not have a body

```
[
    {
        "groceryItem": "Wheat 2 kg",
        "isPurchased": false,
        "_id": "602a853e636d4305f1dcd950"
    },
    {
        "groceryItem": "Rice 5 kg",
        "isPurchased": false,
        "_id": "602a8545636d4305f1dcd951"
    },
    {
        "groceryItem": "Potato 1 kg",
        "isPurchased": false,
        "_id": "602a94c4bd38b103358afb27"
    }
]
```

Request URL:- http://localhost:3000/grocery/getAll

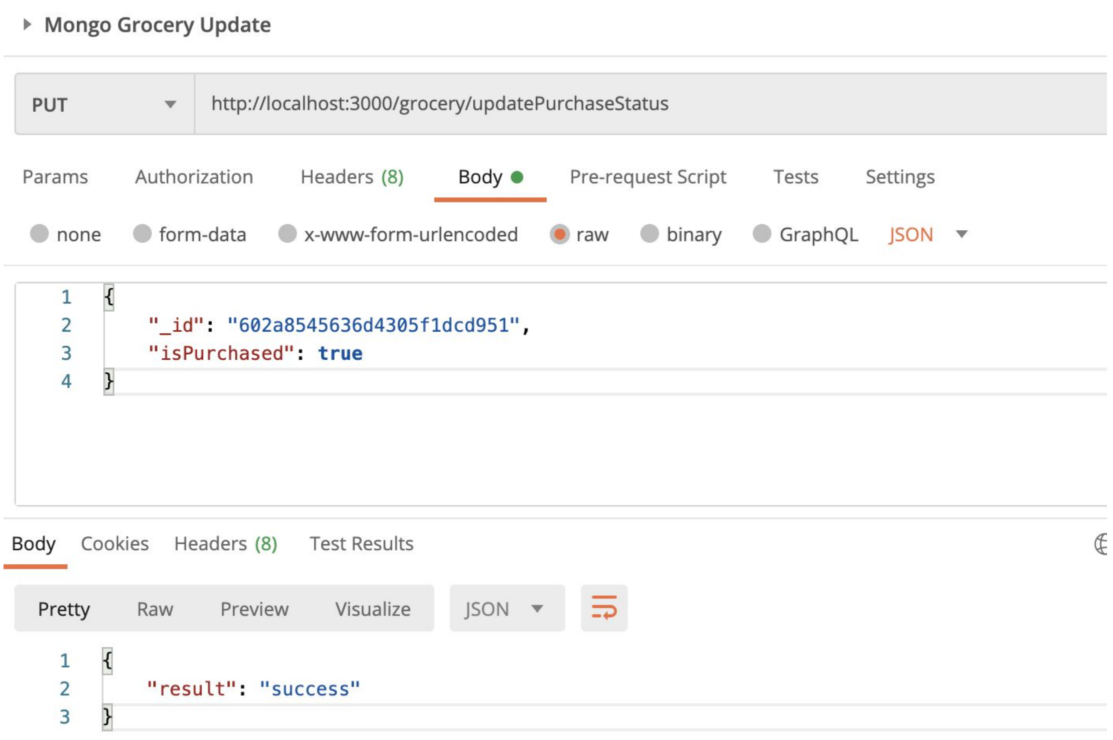
Request Method:- GET

Sample Response JSON:-

```
[
    {
        "groceryItem": "Wheat 2 kg",
        "isPurchased": false,
        "_id": "602a853e636d4305f1dcd950"
    },
    {
        "groceryItem": "Rice 5 kg",
        "isPurchased": false,
        "_id": "602a8545636d4305f1dcd951"
    },
    {
        "groceryItem": "Potato 1 kg",
        "isPurchased": false,
        "_id": "602a94c4bd38b103358afb27"
    }
]
```

Step 3:

Next build an API to update the `isPurchased` value of an individual item by using `_id` value. This API will be used to mark the item as purchased in the website.


Mongo Grocery Update

PUT http://localhost:3000/grocery/updatePurchaseStatus

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
{
    "_id": "602a8545636d4305f1dcd951",
    "isPurchased": true
}
```

Body | Cookies | Headers (8) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "result": "success"
}
```

Request URL:- *http://localhost:3000/grocery/updatePurchaseStatus*

Request Method:- PUT

Sample Request JSON:-

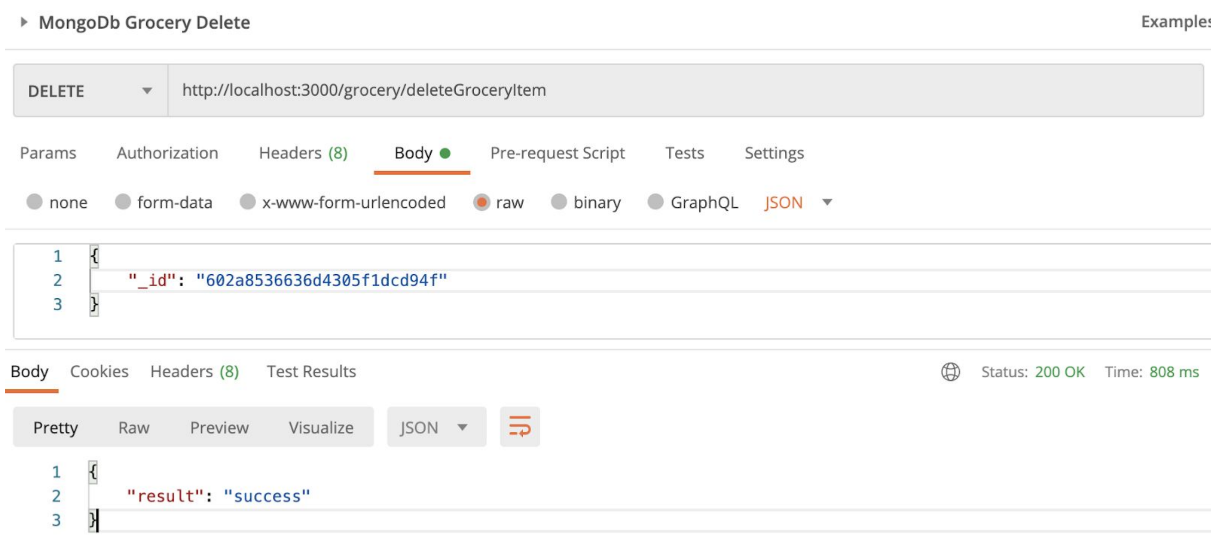
```
{
   "_id": "602a8545636d4305f1dcd951",
    "isPurchased": true
}
```

Response JSON:-

```
{
    "result": "Success"
}
```

Step 4:

Next build an API to delete a grocery item by using `_id` value. This will be used to delete an item when the user clicks on the delete button(x) on the website.


MongoDb Grocery Delete

Examples

DELETE http://localhost:3000/grocery/deleteGroceryItem

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "_id": "602a8536636d4305f1dcd94f"
}
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 808 ms

Pretty Raw Preview Visualize JSON

```
{
    "result": "success"
}
```

Request URL:- *http://localhost:3000/grocery/deleteGroceryItem*

Request Method:- DELETE

Sample Request JSON:-

```
{
   "_id": "602a8545636d4305f1dcd951"
}
```

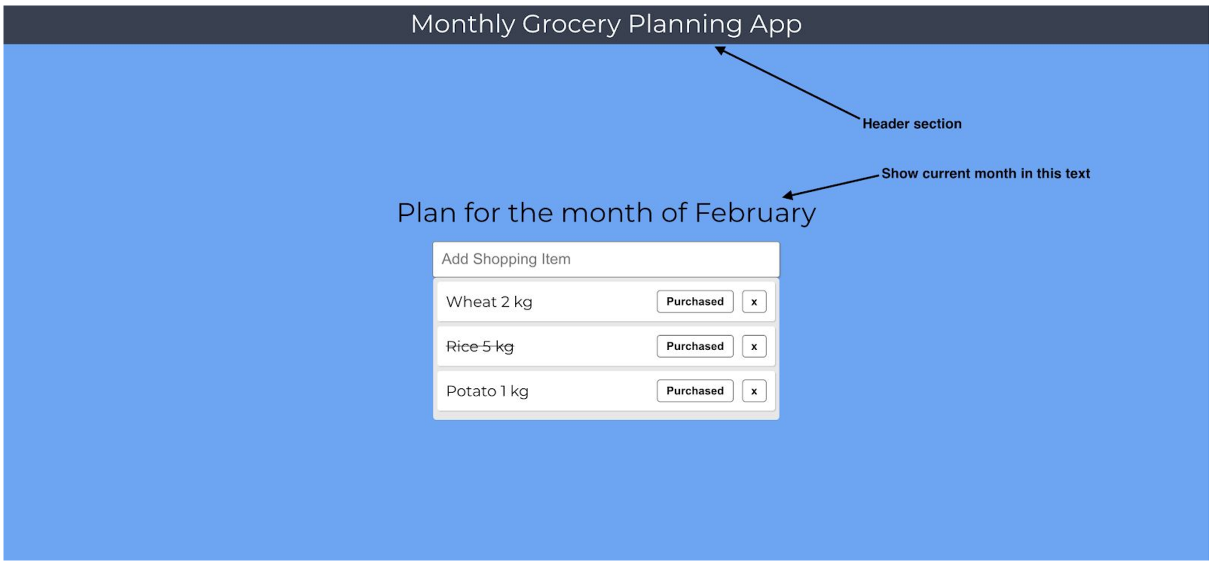
Response JSON:-

```
{
    "result": "Success"
}
```

Frontend development

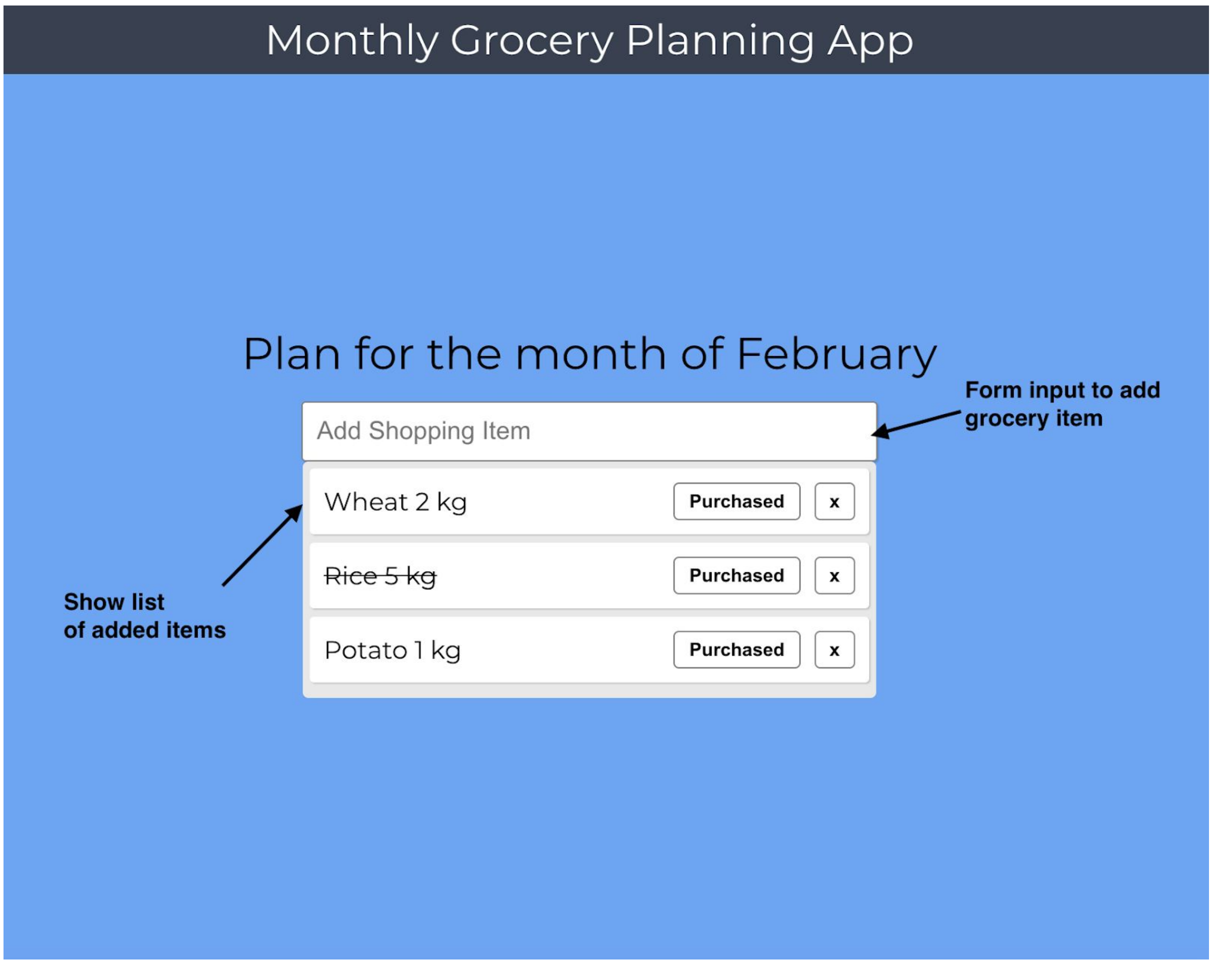
Step 1:

Build the header section and write logic to display current month in website



Step 2:

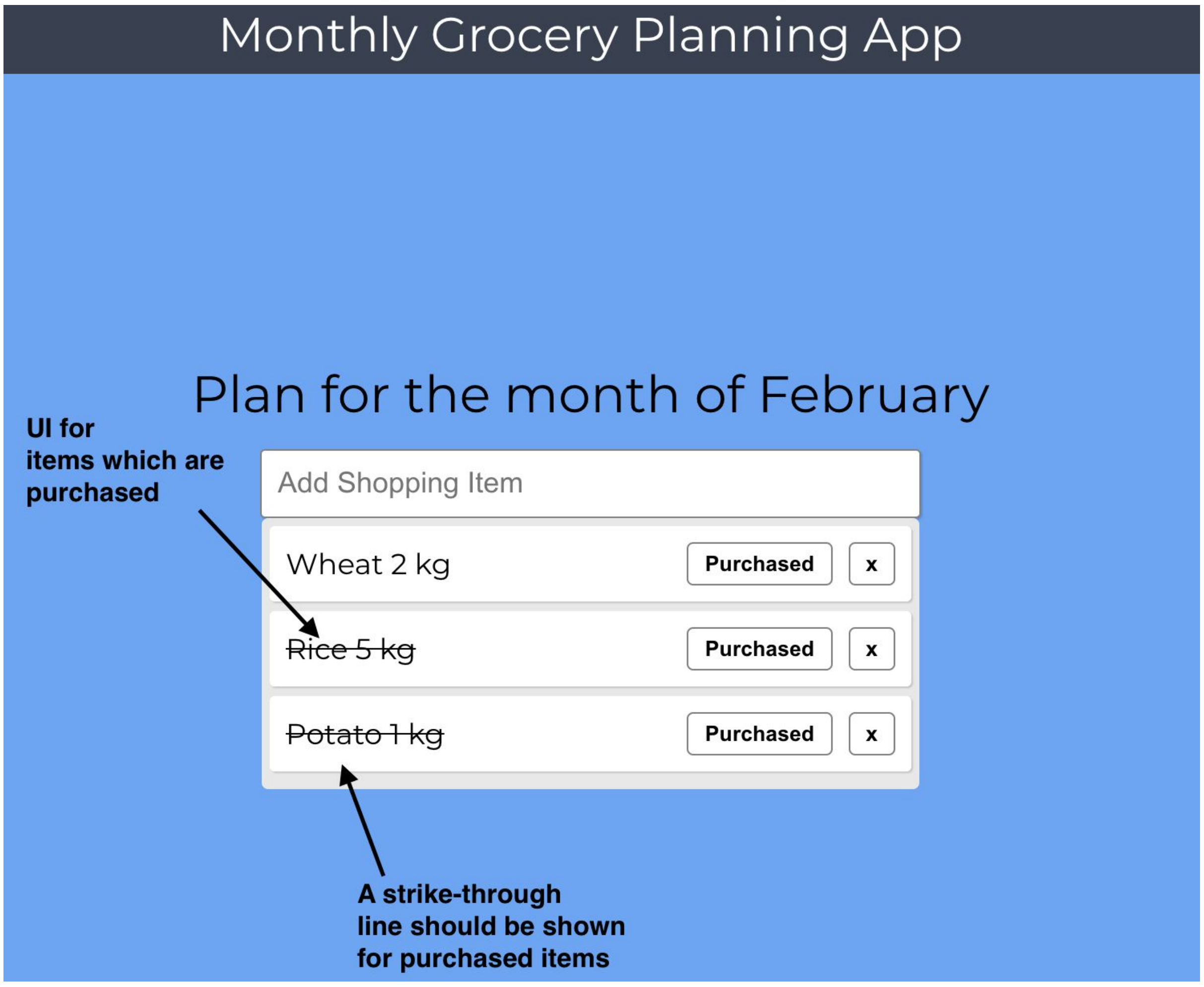
Create a form input to add a grocery item. When a user clicks enter after adding item text, send the item data from the form input element to the `*grocery/add*` backend API for adding item entry to database.
Next get a list of all items from the backend using `grocery/getAll` API and show them below form input.



Step 3:

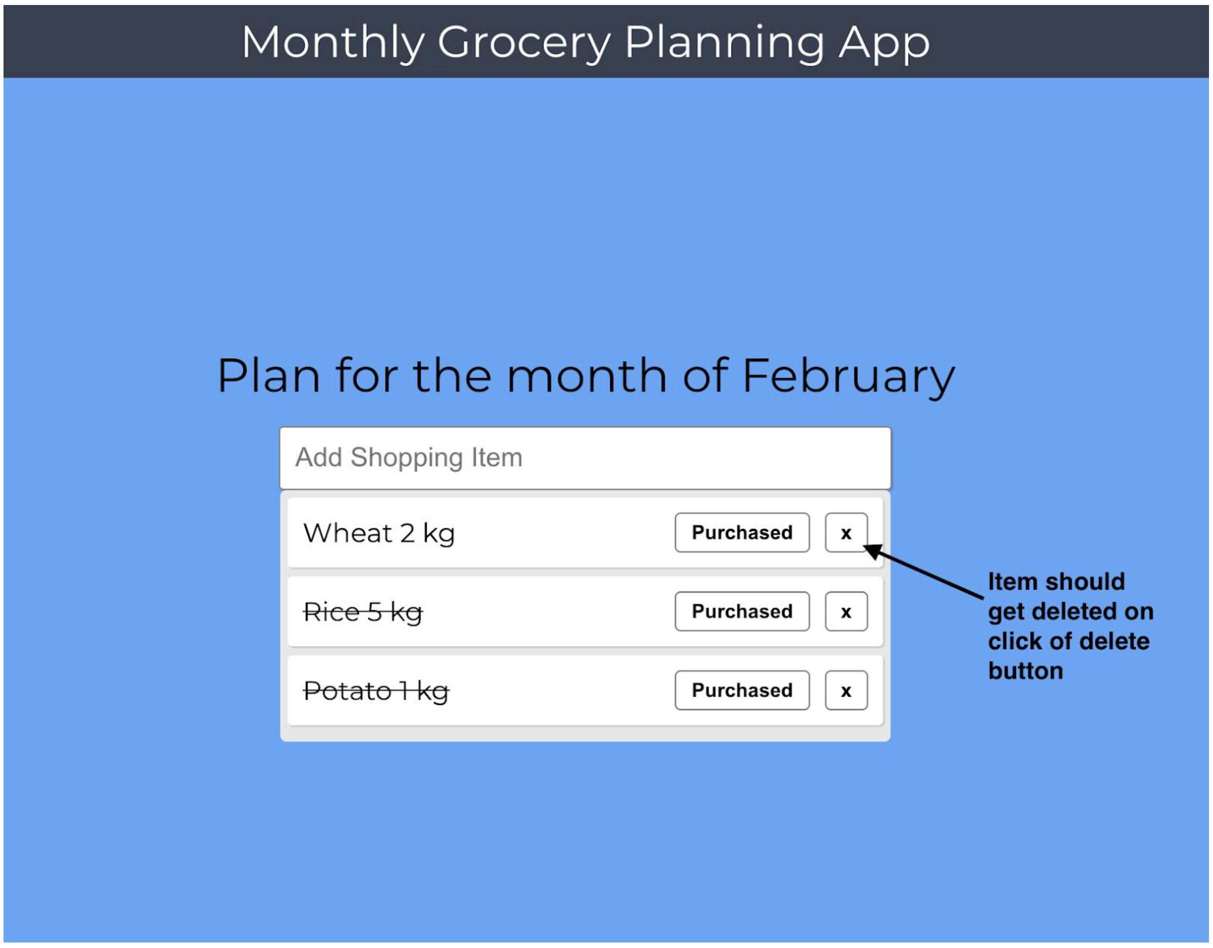
Add a `Purchased` button to each grocery item. When a user clicks on the `Purchased` button, make use of `grocery/updatePurchaseStatus` API to update `isPurchased` entry of grocery item in the backend.

The UI for items whose purchased status is complete is shown below:-


Monthly Grocery Planning App
Plan for the month of February
UI for items which are purchased
Add Shopping Item
Wheat 2 kg
Purchased
x
~~Rice 5 kg~~
Purchased
x
~~Potato 1 kg~~
Purchased
x
A strike-through line should be shown for purchased items

Step 4:

Add an `x` button in the grocery item. On click of the `x` button, make use of `grocery/deleteGroceryItem` to delete a particular item from the shopping list.



Monthly Grocery Planning App

Plan for the month of February

Add Shopping Item

Wheat 2 kg Purchased x

~~Rice 5 kg~~ Purchased x

~~Potato 1 kg~~ Purchased x

Item should get deleted on click of delete button

Learnings:

How to build CRUD apis in express.js and connect the APIs to the frontend website.