

Restaurant Management Website


We are seeking a skilled MERN Stack Developer to develop a full-stack Restaurant Management website. This person will be crucial in redesigning our platform, using MongoDB, Express.js, React.js, and Node.js to create a user-friendly and engaging experience for our customers and staff. The candidate will be responsible for developing features that enhance the restaurant's online presence, improve customer interaction, and streamline internal management processes.

Ensure the Following things to get 100% mark

- Include at least 15 meaningful commits on the client side & 8 meaningful commits on the server side with descriptive messages.
 - Include a README file with the project name, purpose, live URL, key features, and any npm packages you have used.
 - Ensure the website is fully responsive on mobile, tablet, and desktop.
 - Secure Firebase configuration keys using environment variables.
 - Secure your MongoDB credentials using the environment variable.
 - Create a design that encourages recruiters. Color contrast should please the eye & ensure that the website has proper alignment, space and The website does not express gobindo design
-

Deployment Guideline

If your Deployment is not okay you will get 0 and may miss the chance of our upcoming rewards.

- Ensure that your server is working perfectly on production and not throwing any CORS / 404 / 504 Errors.
- Ensure that your Live Link is working perfectly and that it is not showing errors on Landing in your system.
-  ensure that the page doesn't throw any error on reloading from any routes.

- ⚠️ Add your domain for authorization to Firebase if you use Netlify / surge
- ⚠️ Logged in User must not redirect to Login on reloading any private route

Layout & Page Structure

1. Navbar

- Website name/logo
- Home
- All Foods
- Gallery
- Conditional login/logout
- Profile Image. This will only show if the user is logged in, Otherwise show a login button. [Click Here for more details on this](#)

2. **Main Section:** Main Section will show different pages based on routes.

3. **Footer:** A footer with all relevant information and eye-catching design

Authentication System

1. **Login Page:** When you click the login button on the navbar it redirects to the login page. You have to use a password and email-based authentication to log in. The login page will have-

- Email
- Password
- Google login/ GitHub- implement any of one
- A link that will redirect to the Register page

2. **Register Page:** You have to use a password and email-based authentication to register. The Register page will have the following -

- Name
- Email
- photoURL
- password
- A Link that will redirect to the login page

★ For password verification you need to follow this - Must have an Uppercase letter, a lowercase letter, Length must be at least 6 character

- ★ If any of this isn't fulfilled it will show an error message /toast
- ★ After successful login or Register you need to show toast/sweet alert

Home page(public)

- **Banner section:** A slider/banner/ a meaningful section. Inside the banner, there will be a Heading Title, a Short Description, and a button that will redirect the user to the All Foods page.
- **Top Foods section:** Show 6 top-selling Food Items depending on the number of purchases of a food item: (see single food page to know about purchase count) including the following pieces of information:
 - Which data you want to show is up to you.
 - Details Button. On clicking this *button* will navigate to the Single Food Page.
 - See All Button: Below the 6 cards, there will be a see all button that will redirect the user to the All Foods page.
- **Extra section:** Add 2(Two) relevant and attractive sections except the nav, banner, footer, and Top Food section.

All foods page (public)

- **Page Title:** The background and the center of the section will have the page name. [Demo](#)
- **Food Cards:** Here you need to show all the food items stored in your database.
- **Search Functionality:** See challenge part for more info 📌
- **Each Card will have:**
 - Food information from database
 - Quantity (See challenge part for more info 📌)
 - *Details Button.* On clicking this *button* will navigate to the Single Food Page.

Single Food Page (public)


Show details all information about a single food item including the following informations and additionally

- Purchase Count. Indicates the number of time the food was purchased, by default 0
- Purchase button. Clicking this button will redirect users to the food purchase

Food Purchase Page (private)

This page will have a form containing the following information:

- Food Name
- Price
- Quantity
- Buyer Name (Read-only. This will be picked from the logged-in user's information)
- Buyer Email (Read-only. This will be picked from the logged-in user's information)
- Buying Date (This will not be an input field. The current time will be automatically added to the database. You can use `Date.now()` to achieve this)
- **Purchase** button

 **Note:** On clicking the purchase button the information will be stored in the database. Also, On a successful order, you have to show a toast/alert (do not use browser alert).


Gallery Page (public)

- **Page Title:** The background and center of the page should display the title prominently. [Demo](#)
- **Gallery Section:** Show at-least 10 static images added in a gallery section. When a user hover over the image, an overlay will be visible on the image with the user's name and some short description of the image. 💡 For ideas visit this but don't copy this design [link](#). And when the image is clicked the image will be shown in a bigger format similar to a **lightbox**. Use [yet-another-react-lightbox](#) or any other library of your choice to make this

My Foods Page (Private)

In this route, you have to show all food items added by the currently logged-in user (💡 hints: You have to filter data based on the user's email). You can show all the food items in tabular/card. Each row/card will have:

- ➔ Some Food info (example: food img, name, price, etc)
- ➔ an update button/icon.


 clicking the update button/icon will redirect to the update page or open a modal. There will be a form that has the food info and an update button. When clicking the update button the product info will be updated. Don't let other users update your added food items.

Add Food Page (Private)

This page will have a form having the following fields:

1. Food Name
2. Food Image
3. Food Category
4. quantity
5. Price

6. Add By (name & email: this info added from currently logged-in user.)
7. Food Origin (Country)
8. A short description of the food item (ingredients, making procedure, etc.)
9. Add Item Button

 on clicking the add button the information will be stored in the database. On successfully adding a food item, you have to show a toast/alert (*do not use browser alert*)

My Orders (Private)

In this route, show all food items ordered by the logged-in user. You have to filter the data based on the logged-in user email. You can show all the food items on the table/card. Each row/card will have:

1. Some Food info (example: food img, name, price, added time, food owner, etc)
2. Buying date and time. In human readable format. Try to use [moment](#) for formatting date and time
3. a delete button/icon that will help the user delete the ordered item from the list

Challenge Requirements

1. **Food Quantity** - You have to implement a feature on the food purchase page where the feature will be: if the available quantity of a food item is zero then you have to show a message to the buyer that he/she can not buy that item because that *item is not available*. Also, the purchase button will be disabled for the user. Also the buyers can't buy more than the available quantity. (assume that a food item has 20 quantities then a user won't be able to buy more than 20 quantities). Also don't let the user purchase his/her own added food items.
2. **Food Search Functionality:** In **All Foods** page implement a search functionality where foods can be searched using its name.
3. **JWT Authentication:** Upon login, you will create a JWT token and store it on the client side. You will send the token with the call and verify the user.

Implementing 401 and 403 is optional. Ensure you have implemented the JWT token, create a token, and store it on the client side for both email/password-based authentication and social login. You must implement JWT on your private routes.

4. Theme Customization

Add a Theme Toggling Button for changing themes from light to dark / dark to light. By changing the theme, it will make the full system dark / light based on user interaction.

Profile Image Functionalities

On clicking the profile image on navbar show a dropdown menu with 3 links.

1. My Foods
2. Add food
3. My Orders

What to Submit

1. Live Site Link :
2. GitHub Repository (**server**) :
3. GitHub Repository (**client**) :

Optional (But Highly Recommended)

1. Add a spinner when the data is in a loading state. You can add a gif/jpg, use any package, or customize it using CSS.
2. Explore and implement any of the animations from the Framer Motion.
3. Explore and implement Tanstack query mutations in your api.
4. Add multiple filtering systems on all food items pages. The filtering system should be implemented on the server side. Think about MongoDB \$and, \$or operators.
5. Pagination: on all Food pages => At the bottom of the cards section you need to create a pagination. First, you have to load 9 cards of data from

the database and then you will show the others based on the page number. Implement backend pagination.

6. Gallery section: On this page, you can implement infinite image scrolling. Initially, there will be 12 card images. After scrolling down the rest of the image will be shown. Also, add simple animation when the image loads.

Some Guidelines:

1. Do not waste much time on the website idea. Just spend 15-20 minutes deciding, find a sample website, and start working on it.
2. Do not waste much time finding the right image. You can always start with a simple idea. Make the website and then add different images.
3. Don't look at the overall task list. Just take one task at a time and do it. Once it's done, pick the next task. If you get stuck on a particular task, move on to the next task.
4. Stay calm, think before coding, and work sequentially. You will make it.
5. Be strategic about the electricity issue.
6. use chat gpt to generate JSON data. You can use chatGPT for Other purposes as well.