Diagnostic Center Management System - Detailed Project Requirements

assignment12_category_0004

Key rules:

- Include a minimum of 20 notable GitHub commits on the client side
- Include a minimum of 12 notable GitHub commits on the server side
- Add a meaningful readme.md file with the name of your website, Admin username, password and live site URL. Include a minimum of 10 bullet points to feature your website.
- Make it responsive for all devices. You need to make it responsive for mobile, tablet and desktop views. Make the dashboard responsive as well.
- After reloading the page of a private route, the user should not be redirected to the login page.
- Use the Environment variable to hide the Firebase config keys and Mongodb credentials.
- Don't use any Lorem ipsum text in your website.

Objective

Develop a robust Diagnostic Center Management System, a full-stack web application that effectively manages appointments, patient records, test results, and administrative tasks for a diagnostic center.

Requirements

1. User Authentication and Profile Management

- Implement user authentication using Firebase Authentication for secure email/password login.
- Allow users to register with their
  - email
  - name
  - avatar(use imageBB to upload the user avatar)
  - blood group(a selector with option A+, A-, B+, B-, AB+, AB-, O+, O-)
  - district(select option)
  - upazila(select option)
  - password
  - confirm_password

  💡 **status:** While you are saving a user data in DB every user will have a default status **“active”** . Admin can block a user. Then the status for the user will be **“blocked”**. We will talk more about this feature in admin section

- Application will have a login page from where registered users can log in using their email and password.

2.User Dashboard(private 🔒)

- Upon successful login, users are redirected to the user dashboard.

  Here the routes will be [***my profile****,* ***My Upcoming Appointments****,* ***and Test results*** *]*

- **Upcoming Appointments :**
  - Users can view a list of upcoming appointments they have booked.
  - Each appointment displays key details, including the test name, appointment date, and time.
  - Users have the option to cancel appointments.

- **Test Results:**
  - Users access a section to view their test results [**basically all delivered reports**] .
  - Users can download or print test results for their records.

- **My Profile**
  - Here User will see his profile details. He will be able to edit his profile data.

3. Homepage

- There will be a navbar with the necessary routes. Don't show private routes if the user is not logged in. If an admin is logged in he will see the admin dashboard route . If a user is logged in he will see his own dashboard. Try to make a decision about how to implement this.

- Make a banner. This will be dynamic. Admin can upload data from the dashboard. The admin will select which banner to display on the home page. Based on the info display all properties in the banner.

provable info will be **[title, image, text, coupon code, discount rate for coupon code, etc, just display the banner which status "**isActive= true"**] .**

- In the banner, there will be a button to navigate the '**all tests'** page.

- Display all featured tests [featured tests are mostly booked tests by the users],
- promotions [promotions data can be static data] Provide some relevance promotional info from your imagination.

- **Personalized recommendation**: Recommendations may include health tips, preventive measures, or upcoming tests suggested by healthcare professionals. [This data will be fetched from the server] .There will be many recommendations . So try to implement a slider . **[You can implement a system to upload data by admin . Or insert data manually in the DB. It's up to you.]**
- A Footer with relevant info .

4 . All tests page.

- Here all the available tests will be displayed along with proper info . like: image , date, slots , title, short description etc . **[You must display test data available for the future dates, so you need to filter out here.... Suppose today is 11/21/23, so you will only display data starting from date of today.]**

- There will be a search feature to filter by date .

- Each card will have a detailed button . After clicking on the button the user will be navigated to the details page .

5. details page :

- Here users can see the details of the test .
- They can book it using the book now button [if available slot count is greater than 0].
- On booking available slots will be reduced by one.
- When a user book a service, by default the **report** status will be **pending .**
- While clicking the Book now button there will be a popup for payment . **He can apply for a promocode and then the discount rate will be applied on the price . Then he can make payment using stripe.**

6 . ADMIN DASHBOARD

[Note , Here test or service will be considered as same term]

- **All users route**
    - **A**dmin can see all the users in the all users route.
    - Admin can download details of a user in pdf file format. Once

      Admin click on download details button a pdf file will be downloaded **containing all the info of this specific user, including all the tests s/he had booked, the delivery status of the test** . This is a special part for you . You please don't ask for any clue. You are expert enough to deal with this requirement. Do it by yourself . you may use this package [https://www.npmjs.com/package/jspdf] . It's not a big deal if you cannot design pdf well. But ensure all the info are there.

- Admin can see details of a user in a modal after clicking **see info** button of a user.
- Admin can change a user status . **[active or blocked]** . If blocked, this user can not access his dashboard and can not book a test/service.
- An admin can change the role of a user and can make him admin.

- **Add a test route**
  - Here an Admin can add a new test . there will be input fields for test name , image url, details, price, date, slots etc**.**
  - On successful insertion you need to show a confirm toast.

- **All tests route**
  - Here all the tests/services will be displayed in a table format.
  - Admin can delete a test/service data.
  - Admin can update a test/service Data.
  - Admin can see all reservations under a test given below .

- **Reservation**
  - Here the admin can see all the reservations of users for the specific test .
  - There will be a searching system to search reservations of a user by his/her email.
  - He can cancel a reservation
  - He can submit the test result for the reservation. [On submitting the result the **status** of the **report** will be **delivered** . [make your own decision on how to submit the test report, for example You can submit pdf link , doc link etc or implement some other feature .]

- **Add banner**
  - Here admin can upload data for a banner . [name , image , title , description, coupon code name, coupon rate, isActive= false]

- **All Banners**
    - Here all the banner info will be displayed in table format. Admin can delete a banner.
    - Admin can select one and only one banner to display in the Home banner. So he will change isActive= true for a banner , Other banner's isActive status will be false by default.
- **Statistics page**
    - Try to implement any chart based on mostly booked service .
    - Implement chart for service delivery ratio. pending/completed .

Challenge Section

- Try to implement pagination in **ALL tests** page .
- Design 3 extra pages with relevant info [Must be relevant with proper text content . NO hints will be provided . Don't ask about hints . Make your own decision.]
- Implement jwt using localstorage [don't go for cookie]. Try to implement 401,403 error for unauthorized access .
- Implement tanstack query in all the data fetching functionality (For GET method only)

Resources

Use this github repository to find all of the districts and upazilas data:
https://github.com/nuhil/bangladesh-geocode

What to Submit

1. **Assignment Category/variant: **

2. **Admin email:**

3. **Admin password:**

4. **Front-end Live Site Link:**

5. **Client Side GitHub Repository Link:**

6. **Server Side GitHub Repository Link:**