Polling and Survey App with Payment Integration

assignment12_category_0019

A company is seeking a talented Developer to contribute to developing an advanced Polling and Survey application using the MERN (MongoDB, Express.js, React.js, Node.js) stack. This project involves integrating payment functionalities, implementing a robust user management system, and creating an admin dashboard with role management. As a vital team member, you will be pivotal in crafting a feature-rich platform for survey creation, voting, result analysis, and user interaction.

🚩: 0 [If there are any updates, they will be mentioned here]

Main Features 📋

Pages:

- **Homepage 📄🌟 (implement any of 4 sections)**
    - Hero Section 🚀
        - A banner describing the website overview.
        - Can have an explore button
        - Use a background image
    - Featured Surveys Section 🌟
        - Most voted surveys(6 survey)
    - Latest Surveys Section 📅
        - Most recently created survey(6 survey)
    - How It Works Section 🛠️

        - Make this section from your likings and it should match the website theme
    - Testimonials Section 🌟👥
        - You can make this section static
        - Each testimonial should have an avatar and a testimonial
    - FAQ❓📚
        - Add some meaningful FAQ

- **Surveys Page (public):** all surveys will be shown with proper information like title. Sort description, total voted, etc. Users can filter by title, category, and vote.
- **Survey details Page (public):** here will have all the information about the survey. And users can vote in a poll. **pro-user** users can add comments to the poll. This page will be public but only authorized users can vote.
    - Only logged-in users can participate in surveys. Make sure only the **user** and **pro-user** roles can participate in a survey.
    - A user is not able to participate in a survey twice. Ensure the prevention of duplicate submissions from the same user.
    - Only **pro users can** comment in a survey. Show the comments that are added by **pro-user** users(but keep in mind any **user can see all comments**).
    - Display survey results visually by charts. After they vote or the deadline expires
    - Allow users to view survey results with charts after the survey deadline or after voting to prevent further votes.
    - Enable users to like or dislike a survey.
    - Allow users to report a survey for inappropriate content.

- **Pricing page (public):**
    - Integrate a payment system for users to be **pro-user** members.
    - Add a **pro** nav link to the navigation bar on the home page from where users can redirect to become a pro user page and able to become a pro user member while the user pays successfully.
    - The role of the user will be changed to the **pro-user** role on successful payment.

User Authentication 📤

- Allow users to create accounts with email and password.
- Implement social media authentication.
- Generate and store JWT tokens on the client side for both email/password-based authentication and social login.
- Implement JWT on private routes with optional 401 and 403 handling.

Role 🧑‍💼

- Create user roles (e.g., user, surveyor, admin, **pro-user**) with different permissions.
- By default, the newly created user will be a user role.

Dashboard

Surveyor Dashboard

Survey Creation 📝

- Able to create surveys with various question types from his dashboard survey creation page.
- A survey will contain the following information.
    - A title.
    - Description.
    - Options. (Yes or no).
    - Like or dislike. Initially like and dislike will be 0.
    - Category. You can search for the most common survey category from Google or chatGPT.
    - Timestamp. (add this from the backend).

Access Control (dashboard) 🔒

- Implement role-based access control to manage permissions effectively.
    - Admin:
        - Manage users.
            - Can change user roles to admin/surveyor.
        - Make a filter system to show users based on **pro-user**, normal users, and surveyor roles.
        - Publish or unpublish survey status.
        - When unpublish a survey make sure the admin will give a feedback message.
        - Can see all payments of pro-user members.
        - Survey responses with
            - A table (name, email, time, voted)

        - Chart
  - Surveyor:
    - Can create or update a survey.
    - Can see all feedback for individual surveys given by users. (Use modal to show a message)
    - Can see feedback messages for individual surveys given by the admin on unpublish. (Use modal to show a message)
    - Survey responses with
      - A table (name, email, time, voted)
      - Chart
  - User:
    - Can participate in a survey.
    - Like or dislike a survey.
    - Can report a survey.
  - **pro-user**:
    - Will have all permissions of user role
    - In addition, pro users can comment on a survey.

Bonus Requirements 🌟

Readme 📝

- Create a readme with at least 5 bullet points for the client.
- Create a readme with what challenges you have faced for the backend.

- Include the live link to the client-side application in the readme.

Reload 🔄

- Ensure that reloading protected/private routes (after login) does not redirect the user to the login page but keeps them logged in.

Responsive Design 📱💻

- Make the application responsive for various mobile, tablet, and desktop devices.

Environment Variables 🔐

- Use environment variables to securely store API keys, sensitive configuration information, and payment gateway details.

Package (implement any of 3)

- Transtack Query / SWR
- Headless UI / shadcn/ui
- Yup
- Mongoose
- moment

Optional

Extra Pages Recommendations 📄

- About Us 🏢
- Contact Us 📞✉️
- Privacy Policy and Terms of Service 🕵️🔒

- Help Center 🆘📖

Remember to maintain a consistent design and user experience throughout your website. Tailor these additional features and recommendations to suit the unique characteristics and goals of your Polling and Survey application. 🚀✨

Guidelines 📌

- Spend 15-20 minutes deciding on the core features of the survey application.
- Start with a basic idea and progressively add more features.
- Prioritize user experience, data security, and seamless payment integration.
- Use ChatGPT for generating sample data initially and adapt as needed.
- Regularly commit and update the readme as you progress through the development stages.

What to Submit

1. **Assignment Category:** assignment12_category_0019
2. **Admin email:**
3. **Admin password:**
4. **Front-end Live Site Link:**
5. **Client Side GitHub Repository Link:**
6. **Server-Side GitHub Repository Link:**