Technical Assessment.

Technical Assessment Instructions:

1. Please read the instructions carefully before proceeding with the assessment.
2. The assessment is designed to test your technical skills and knowledge in the relevant programming language(s) and skills.
3. You will have a maximum of 5 days from the moment you have received the email , to complete the assessment.
4. The assessment will consist of real-world scenarios
5. Submit your solution(s) in a separate file(s) along with clear documentation on how to run your code.
6. Your solution(s) will be evaluated based on correct functionality, code quality, and design.
7. Good luck and show us your skills!

Note: Failure to follow the instructions may result in partial or complete disqualification of your assessment results.

The Test

Build a Multitenant TODO list application that allows a user to create and manipulate multiple to-do lists and invite other users to collaborate on specific lists.

Features:

- User authentication and authorization
- Ability for a user to create and manage multiple to-do lists
- Ability for a user to invite other users to collaborate on specific lists
- Ability for users to mark to-do items as complete or incomplete

Optional Features:

- Ability for users to prioritize to-do items
- Ability for users to search and filter their to-do lists

Technical Requirements:

- Multi-tenancy architecture to support multiple users and their separate lists
- Data storage and management for user accounts, to-do lists, and to-do items
- User authentication and authorization using a secure method
- Ability to send email notifications (for invitations and whatever else you think needs email notifications)
- Responsive design for web and mobile access

Technologies to use:

- React.js for the frontend
  - React Router to handle navigation
  - Axios for API requests
  - Whatever UI library you want to use, we use mostly Bootstrap or Tailwind, but it doesn't matter and you can write your own CSS if you prefer.
- Node.js for the backend
  - Express.js to create your API
  - MongoDB with Mongoose for an ORM

Deployment (Instructions on where to host for free if you don't want to spend money on a VPS or paid hosting):

- Use netlify or Vercel to deploy the frontend, they are both free.
- Deploy the backend on Cyclic (cyclic.sh) it's free.
- Use Mailtrap (mailtrap.io) for emails they have a free tier with 1000 free monthly emails and a dashboard to test if your emails work and instructions to use nodemailer to send emails so your integration is just 1 function.
- MongoDB has a free Database that you can use for deployment, if you've already used that, Railway (railway.app) has a free tier you can use too.

Notes:

- There's no need to create custom email templates, just plain text will do for the demonstration
- Use version control (preferably store your code on GitHub) during your build process and make sure you use clear commit messages.
- Have a Readme.md for all your repositories with instructions for when we want to run and test your code locally.

Submission:

- Send us:
  - The link to the hosted frontend
  - Link(s) to the GitHub repositories.
- If you need any extra accommodation, please feel free to react out to us.