Hey, thanks for agreeing to take up the coding exercise! This exercise will give us a good idea about your coding style and approach to building software. It would also form the basis of our future discussions in the hiring process. Please go through it and let us know the date by when you would submit the solution.

Problem Statement

Your team is implementing an *Enterprise Form Builder* that allows select administrators to define and publish forms that have to be filled and submitted by employees for different purposes. Example use cases: leave application form, employee NPS survey, etc

You've been entrusted to design and implement the backend of this Form Builder app, with appropriate ReSTful or GraphQL APIs.

Form Structure

Form Title

This can be used to make the form easily identifiable by the admins for form management and by the employees once the form is made available for them to use.

Form Field

This is the core component of a form through which data collection can happen. A form can have one or more fields. Fields can be configured at varying degrees depending on the type and context.

Common Field Properties

Field Label

This is a short text to give a name to the field. Every field should have a label.

Is Required/Mandatory?

Admin can decide whether a field is mandatory or not, i.e. whether or not the form can be submitted when there's no value entered in the field.

Field Types

Basic/Primitive Fields

1. *Text*
    - Accepts all characters
    - Configuration options:
        - Maximum Length
        - Minimum Length
        - Restriction on allowed characters and their patterns - can be specified via regular expression

2. *Number*
    - Accepts only those characters that can represent a numerical value
    - Configuration options:
        - Maximum Value
        - Minimum Value
        - Whether strictly an integer, or floating point allowed
        - If floating point, then it should be mandatory to specify the maximum number of decimal places allowed

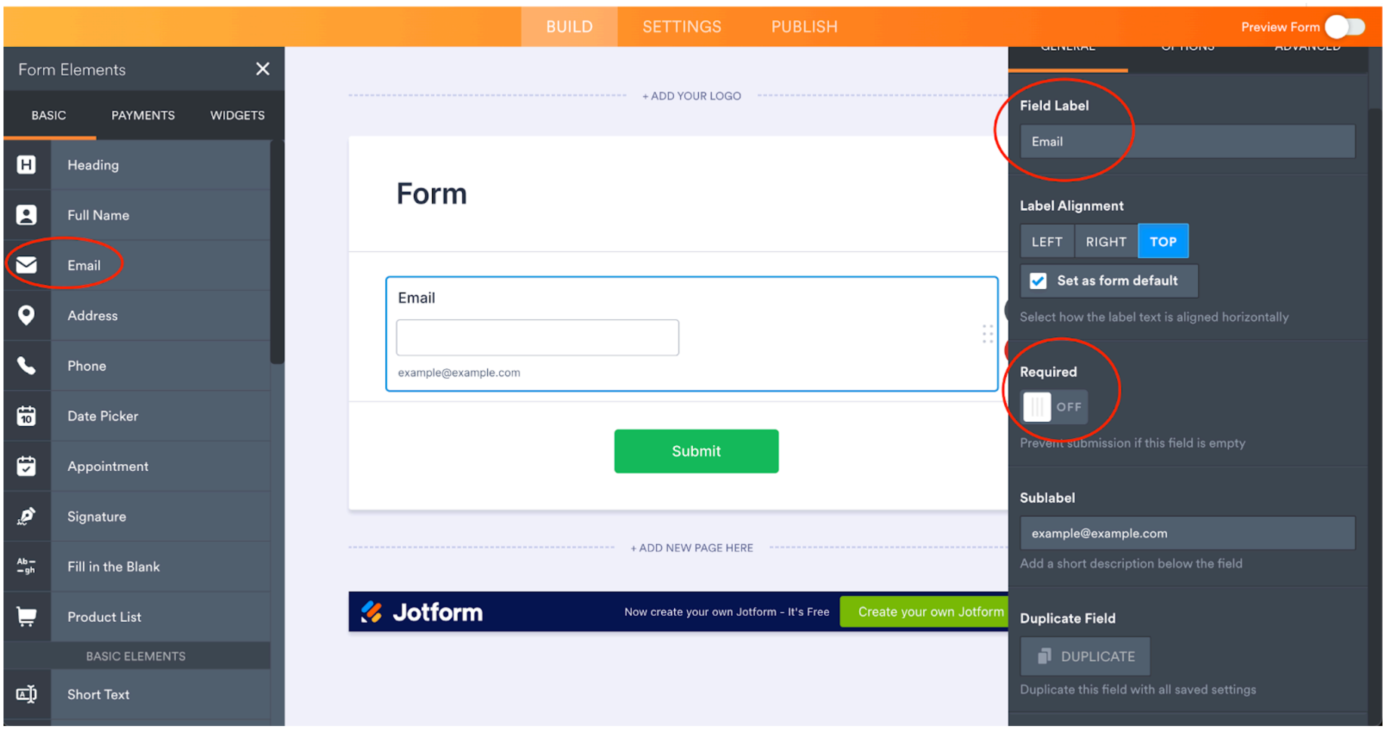
Submit Button

Data entered in the form fields can be submitted by an employee by clicking on this button. The form submission service should not accept the data if there's any validation failure, and should return all the validation errors in a way that the UI can display them all appropriately for the submitter to make the necessary corrections.

The admin can only configure the label of the submit button

UI Example

You can refer to JotForm as an example of how a Form Builder interface might look like: https://www.jotform.com/myforms/ (elements relevant to this exercise are encircled in the screenshot)



Email

example@example.com

Field Label: Email

Label Alignment: LEFT RIGHT [✓ TOP]

[✓] Set as form default

Select how the label text is aligned horizontally

Required: OFF

Prevent submission if this field is empty

Sublabel: example@example.com

Add a short description below the field

Duplicate Field

Duplicate this field with all saved settings

Exercise Requirements/Expectations

1. Your solution must build+run on Linux or Mac OS.
2. Programming Language: Kotlin (preferred) or Java
3. Database: PostgreSQL preferred. Or you can use dummy data, but do maintain appropriate separation of concerns.
4. Your code will be evaluated for both functional and non-functional requirements such as code quality, ease of setting up dev environments, test execution and readability, etc.
5. Automated tests are mandatory, so please include tests/specs. Additionally, it's a huge plus if you test drive your code - try to reflect that in your code commits.
6. We are really, really interested in a clean and maintainable codebase, so please solve the problem keeping this in mind.
7. When implementing this solution, please use Git for version control. We expect you to send us a zip/tarball of your source code when you're done; that includes Git metadata (the `.git` folder) in the tarball so we can look at your commit logs and understand how your solution evolved. With this approach, there's no need to publish to a cloud based service like Github.
8. Please include instructions (e.g. a ReadMe file) for building+running the code and executing the tests – preferably a single command/task for setting up dependencies and services, running the app, executing the tests, etc
9. Please do not make either your solution or this problem statement publicly available by, for example, publishing as a public repository of a cloud service (Github, Bitbucket, Gitlab, etc) or by posting to a blog, forum, etc.
10. Authentication and authorization are out of scope. You can assume that the user already exists.

Do let us know if you have questions at any time during the assignment by writing to hiring-tech@kodo.in

Good luck!