PHP Interview Task – Dynamic Form Builder

Interview Task – Simple Explanation

- **Admin can create multiple forms dynamically (no fixed / hardcoded forms).**
- **Each form can have any number of fields (text, dropdown, checkbox, etc.).**
- **These forms and their fields are stored in the database.**

Front-End (User Side)

- **All the forms created by the admin will be listed on the front end.**
- **User can select any form from the list.**
- **The selected form will be generated dynamically from the database.**
- **User fills the form and submits it.**

Data Storage

- **User's submission will be stored in the database, field by field.**

- **The structure should support any form with any field combination.**

==========

Admin Side (View Responses)

- **Admin can select a form.**
- **Admin can see:**
    - **List of submitted responses**
    - **Each submission's field-wise values**
- **Admin should be able to clearly view which user submitted what data.**

Key Rule (Important for Candidates)

❗ No form or field should be hardcoded
❗ Everything must be created, rendered, and stored dynamically using DB tables

---

MODULE 1: ADMIN – CREATE FORMS

Step 1: Create a Form

Admin creates a new form by entering a Form Name.

Example:

- **Job Application Form**
- **Feedback Form**

This is stored in the `forms` table:

forms
----
id | form_name | created_at

Step 2: Add Fields to the Form

Admin can add multiple fields to a form.

Each field contains:

- **Field Label (e.g., Full Name)**
- **Field Type**
  - **text**
  - **textarea**
  - **number**
  - **date**
  - **dropdown**
  - **checkbox**
  - **radio**
- **Required (Yes / No)**
- **Placeholder (optional)**
- **Sort Order (field position)**

Stored in:

form_fields
----
id | form_id | label | type | required | placeholder | sort_order

---

Step 3: Add Options (Only for Dropdown / Radio / Checkbox)

If the field type is:

- **dropdown**
- **radio**
- **checkbox**

Admin must enter options.

Example:
Gender → Male, Female, Other

Stored in:

field_options
----
id | field_id | option_text

Step 4: Save the Form

On save:

- **One entry is added to `forms`**
- **Multiple entries are added to `form_fields`**
- **Options are added to `field_options`**

✅ **The form structure is now fully saved in the database.**

MODULE 2: USER – FILL FORM (DYNAMICALLY GENERATED)

Step 1: View Available Forms

User sees a list of forms pulled from the database.

Example:

- **Job Application Form**
- **Feedback Form**

User clicks a form →
`form.php?form_id=3`

Step 2: Load Form Data from Database

Fetch form details:

```
SELECT * FROM forms WHERE id = 3;
```

Fetch fields:

```
SELECT * FROM form_fields
WHERE form_id = 3
ORDER BY sort_order;
```

If field type is dropdown / radio / checkbox:

```
SELECT * FROM field_options WHERE field_id = FIELD_ID;
```

Step 3: Build the Form Using PHP (Dynamic Rendering)

Examples:

Text Field

<input type="text" name="field_5" placeholder="Enter Name" required>

Dropdown

```
<select name="field_12">
  <option>Male</option>
  <option>Female</option>
</select>
```

✔ Add `required` attribute when needed
✔ No HTML fields should be hardcoded
✔ Form is fully generated using database data

Step 4: Submit the Form

When the user submits:

Store main submission

form_responses

```
---------------------------------
id | form_id | user_id | submitted_at
```

Store each field value

form_response_values
----
id | response_id | field_id | value

Example:

- **Full Name → John Smith**
- **Gender → Male**

This structure supports any number of forms with any field structure.

MODULE 3: ADMIN – VIEW FORM RESPONSES

Step 1: Select a Form

Admin selects a form and sees all submissions.

SELECT * FROM form_responses WHERE form_id = 3;

Step 2: View Individual Submission

For each submission:

- **Fetch field labels from `form_fields`**
- **Fetch values from `form_response_values`**

Output Example:

Field Name	Value

Full Name	**John Smith**
Gender	**Male**
Experience	**3 Years**

USER FLOW (Summary)

1. **View list of forms**
2. **Select a form**
3. **Form loads dynamically from DB**
4. **User fills and submits**
5. **Data is saved in normalized tables**

ADMIN FLOW (Summary)

1. **Create form**
2. **Add fields and options**
3. **Save form**
4. **View submissions**
5. **View individual responses**

What This Task Evaluates

- **PHP logic & loops**
- **MySQL database design**
- **Dynamic HTML generation**
- **Normalized data storage**

- **Clean backend thinking**

NOTE :

- **All forms and fields must be fully dynamic. No form, field, or input should be hardcoded anywhere in the code.**
- **Server-side validation is mandatory for all required fields. HTML validation alone is not sufficient.**
- **For dropdown, checkbox, and radio fields, values must be validated against the options stored in the database.**
- **User identity can be assumed as a dummy logged-in user ID. Full authentication is not required for this task.**
- **Database structure must support any number of forms with any number of fields without schema changes.**
- **UI can be simple (Bootstrap or basic CSS is enough). Logic and functionality are more important than design.**
- **Code should be clean, readable, and modular (separate DB connection, logic, and view files).**
- **Expected completion time: at the earliest.**
- **Submission must be shared as a Git repository link**