1. What is an HOC (Higher-Order Component) in React?

- An **HOC** is a **function** that takes a component as input and returns a **new component** with additional props, logic, or behavior.
- It's often used for **code reuse** (e.g., authentication checks, logging, theming, data fetching).

Example idea (not code yet):

- You have a simple component `UserProfile`.
- You wrap it with `withAuth(UserProfile)` → this new component checks if the user is logged in before rendering `UserProfile`.

2. Why use HOCs?

- To **avoid duplication** of common logic.
- To separate **concerns** (UI logic vs. business logic).
- To make components **more reusable**.

3. Implementing a Simple HOC

Steps:

1. Define a function (HOC) that accepts a component.
2. Inside the function, return a **new component** that adds behavior.
3. Use this wrapped component instead of the original one.

Example use case:

- A HOC that logs when a component is mounted/unmounted.
- A HOC that provides loading indicators.
- A HOC that injects additional props (like theme or data).

📌 Case Study: Feedback Form with HOC

🎯 Objective

We want to build a **Feedback Form** application where employees can share their feedback (like *"MileStone2 Completed – Yes/No"* or *"Training quality"*, etc.).

Instead of duplicating logic (like input validation, logging, or authentication) across different forms, we will use **Higher Order Components (HOCs)** to manage these **cross-cutting concerns** in a reusable way.

🏗️ Application Flow

1. A user (employee) opens the **Feedback Form**.
2. The form contains:
   - Text input for comments
   - Options (Yes/No, ratings, or multiple-choice questions)
3. Before submission:
   - Validation ensures no empty input.
   - User identity is checked (whether logged in).
   - The feedback is logged for analytics.
4. Finally, the feedback is submitted.

⚡ Why Use HOCs Here?

- **Validation logic** → Can be reused across different forms (Feedback Form, Registration Form, Survey Form).
- **Authentication check** → Ensures only logged-in employees can submit.
- **Logging/Analytics** → Tracks how users interact with forms.

If we directly add this logic into each form, code becomes repetitive and messy.
👉 With HOCs, we **wrap the FeedbackForm** with reusable logic.

🔑 HOCs We Will Use

1. withValidation

- Adds validation rules to the form.
- Checks if all required fields are filled.
- Prevents submission when validation fails.

Example behavior:

- If employee tries to submit blank feedback → "Please fill out this field" message appears.

2. withAuthentication

- Ensures that only logged-in employees can access the form.
- If the employee is not logged in → Redirect to login page.

Example behavior:

- If someone outside the company tries to open the form, they cannot access it.

3. withLogging

- Tracks employee interactions with the form.
- Logs actions like *“Form Opened”*, *“Submit Clicked”*, *“Validation Error Occurred”*.
- Useful for admin to analyze form usage.

Example behavior:

- Admin can later check how many employees submitted “Yes” vs “No” for Milestone2.

🖼️ Putting It All Together

Think of it as **layers wrapped around the FeedbackForm**:

```
Employee submits Feedback →
   withAuthentication (check login) →
   withValidation (check fields) →
   withLogging (track actions) →
   Actual Feedback Form Submission
```

So the **FeedbackForm** itself only contains the **UI (fields, buttons)**, while HOCs handle the **extra responsibilities**.

📌 Example Scenario

Case: "MileStone2 Completed – Yes or No"

- Employee **Ravi** logs in to the portal.
- Opens the Feedback Form.
- Selects **Yes**.
- Submits the form.

What happens behind the scenes?

1. **withAuthentication** → Confirms Ravi is logged in. ✅
2. **withValidation** → Ensures Ravi selected Yes/No (not left blank). ✅
3. **withLogging** → Records: *“Ravi submitted Yes at 10:45 AM”*. ✅
4. **FeedbackForm** → Saves Ravi’s answer into database. ✅

If Ravi was **not logged in** → He would be redirected to login.
If Ravi **did not select Yes/No** → Validation error stops submission.