3. Developer Notes / What to Regulate

Backend Developers

- **API Contract Stability:** Endpoints must return consistent response structures (JSON fields, types).
- **Validation & Errors:** Provide detailed validation messages; return standard status codes (400, 401, 500).
- **Authentication:** Token-based auth must be consistent; token expiration and refresh should work reliably.
- **Data Integrity:** CRUD operations must be consistent; repeated requests should not corrupt data.
- **Test Environment:** Support predictable test data (bookings, users) for QA to validate frontend behavior.

Frontend Developers

- **API Consumption:** Properly handle API responses, including error messages.
- **Token Handling:** Store and use authentication tokens securely.
- **State Management:** Update UI state to match API data accurately (booking lists, new bookings).
- **Error Display:** Show backend validation errors clearly to users.
- **Async Handling:** Wait for API calls to finish before updating UI or asserting tests.

Both Teams

- Agree on **standardized response formats** and error messages.
- Use **consistent IDs, timestamps, and predictable data** to avoid flaky tests.
- Keep integration **endpoints decoupled from production data**, use test data or mocks.

4. Interview Prep: Integration Testing Questions

Q1: What is integration testing?
A1: Verifies that different modules/components (frontend, backend, APIs, databases) work together correctly.

Q2: Why is API → Frontend integration testing important?
A2: Ensures that changes in API data are correctly reflected in the UI and that error handling works across layers.

Q3: How do you validate authentication tokens in integration tests?
A3: Obtain a token via API, use it in frontend requests, and assert success/failure responses.

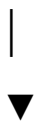
Q4: How to handle errors in integration tests?
A4: Trigger errors via API or invalid input and assert proper handling/display on frontend.

Q5: What should developers regulate to support integration testing?
A5: API response formats, error messages, authentication, data consistency, predictable test data, and frontend state updates.

#	Test Name / Scenario	Input	Expected Result	Actual Status	Type of Test	Positive / Negative	Notes / Theory
1	API → Frontend Data Flow	Create booking via API, fetch data on frontend	Booking created visible on frontend, data	✅ Booking created via API, Data consiste	Integration / Functional	Positive	Verifies that API changes reflect correctly on

			matches API	ncy verified			frontend.
2	Authentication Token Flow	Request token, use token to access endpoints	Token obtained, authentication successful	✅ Token obtained, flow verified	Security / Integration	Positive	Ensures token-based auth works from API to frontend.
3	Data Consistency Across API Operations	Check booking count, create booking, verify count	Booking count increases by 1, data consistent	✅ Initial: 112, New: 113, Final: 113	Integration / Functional	Positive	Validates CRUD operations consistency across frontend + backend.
4	Error Handling Integration	Trigger validation error via API	Proper error message returned and displayed	✅ Validation errors handled	Integration / Functional	Positive	Ensures backend errors are propagated and handled properly on frontend.

Start Integration Tests



[API Booking Creation]

- Send POST request to create booking

- Receive booking ID

▼

[Frontend Data Check]

- Fetch booking list via frontend

- Verify data matches API

|

▼

[Authentication Token Flow]

- Request token via API

- Use token to authenticate frontend actions

- Verify success

|

▼

[Data Consistency Operations]

- Get initial booking count

- Create new booking

- Verify booking count updated

|

▼

[Error Handling Flow]

- Trigger validation errors via API

- Ensure frontend displays proper error messages

|

▼

End Integration Tests

- All assertions passed