Frontend Practice Hackathon 3: Comprehensive Guide

To ensure a smooth and effective learning experience, this document outlines every detail of the Practice Hackathon. Participation in the Practice Hackathon is **mandatory** as it prepares students for the upcoming Hackathon by ensuring all prerequisites are completed. This will help avoid confusion among the 30,000 students participating. The Practice Hackathon focuses exclusively on the **frontend functionality** and will use the same UI/UX templates developed in the last hackathon, allowing students to update or modify the templates as needed. The **UI/UX part will not be evaluated.**

Contents

Frontend Practice Hackathon 3: Comprehensive Guide 1

1. Fake JSON API 2
    - Steps to Create a Fake JSON API: 2
    - Guidelines for Creating Data Schema: 2
    - Best Practices for Data Structure: 3
    - Generate Mock Data: 3
2. Practice E-Commerce API for Frontend Hackathon 5
3. Shipment and Tracking Platform 5
    - Introduction to ShipEngine: 5
    - Key Features of ShipEngine: 5
    - Steps to Integrate ShipEngine: 5
    - Practical Use in Hackathon: 7

FAQs 7

1. Fake JSON API

While the actual API will be provided on Hackathon Day, in the Practice Hackathon, you are required to create and understand API data schemas using platforms like **MockAPI.io** or other fake JSON data generators. This exercise is critical to understanding the data requirements and preparing for the Hackathon.

Steps to Create a Fake JSON API:

1. **Visit MockAPI.io:**
    - Create an account and log in.
    - Start a new project for your mock API.
2. **Define Your Schema:**
    - Add resources for **inventory**, **shipment**, **tracking**, and **sales data**.
    - Include fields like product name, description, price, tags, sizes, images, shipment status, and tracking numbers.

Guidelines for Creating Data Schema:

- **Understand Business Requirements:**
    - Review the project goals and identify the key data types required (e.g., products, users, orders, etc.).
    - Map the business process to data needs. For example:
        - Inventory should include fields like product_name, price, and stock_quantity.
        - Shipment tracking should cover fields like tracking_id, delivery_status, and estimated_delivery_date.
- **Field Selection Best Practices:**
    - Include **essential fields** (e.g., ID, name, price) to support basic functionality.
    - Add **optional fields** (e.g., discount, ratings) for enhanced features.
    - Use **consumer mobile phone numbers** as unique IDs to ensure real-world alignment and easy integration.
    - Ensure all fields are descriptive and aligned with their purpose.
- **Validate Correctness:**
    - Use tools like **Postman** or **MockAPI.io Preview** to review the API responses.
    - Confirm that the data format matches the schema you've defined.

Best Practices for Data Structure:

1. **Segregate Data Logically:**
   - Use separate resources for different data types (e.g., Products, Orders, Users).
   - Avoid duplicating data across resources.
2. **Use Consistent Naming Conventions:**
   - Follow standard naming conventions like snake_case or camelCase for field names.
   - Ensure all field names are meaningful and easy to understand.
3. **Optimize for Frontend Usage:**
   - Ensure data structures are simple and directly usable by the frontend team.
   - Avoid nesting fields unnecessarily to prevent complex queries.
4. **Documentation:**
   - Document your schema for clarity and future reference.
   - Include field descriptions and sample data in your documentation.

Generate Mock Data:

1. Populate your schema with realistic sample data.

Example for Products:

```
{
  "id": "1234567890",
  "name": "Wireless Mouse",
  "description": "Ergonomic wireless mouse with adjustable DPI.",
  "price": 25.99,
  "tags": ["electronics", "peripherals"],
  "sizes": ["Small", "Medium"],
  "image": "https://example.com/image.jpg",
  "rating": 4.5,
  "stock_quantity": 50
}
```

Example for Sales Data:

```
{
  "sale_id": "9876543210",
  "product_id": "1234567890",
  "quantity_sold": 3,
  "sale_price": 23.99,
  "date_of_sale": "2025-01-01",
  "customer_id": "1000000000"
```

```
}
```

Example for Shipment and Tracking Data:

```
{
  "tracking_id": "SHIP12345",
  "order_id": "9876543210",
  "shipment_status": "In Transit",
  "estimated_delivery_date": "2025-01-05",
  "carrier": "FedEx",
  "shipment_origin": "New York, NY",
  "shipment_destination": "San Francisco, CA",
  "customer_phone": "1000000000"
}
```

Example for Inventory Data with SKUs:

```
{
  "sku": "WM-001-BLK",
  "product_name": "Wireless Mouse",
  "stock_levels": [
    {"size": "Small", "color": "Black", "quantity": 30},
    {"size": "Medium", "color": "White", "quantity": 20}
  ],
  "price_adjustment": {
    "discount": 10,
    "faulty_item": true,
    "replacement_policy": "30 Days"
  },
  "last_updated": "2025-01-01T10:00:00Z",
  "supplier_contact": "1234567890"
}
```

2. Ensure all data entries are aligned with the defined schema.
3. Test API Responses:
    - Use tools like Postman to validate API responses and confirm they match the expected structure.
4. **Iterate:**
    - Regularly update and refine your schema as new requirements are identified.

---

2. Practice E-Commerce API for Frontend Hackathon

Resources:

- **Practice API Documentation:** (Prepared By Sir Ali Jawwad)
  https://alijawwad001.atlassian.net/wiki/external/OWVkZTc1NjExMjU1NDljNmFkMGJhZTUxYmVmMWNmMzc
- **Practice Template** (Recommended By Sir Ali Jawwad)
  https://www.figma.com/design/k28EtUgXHEtIpqZz9Zhv11/Full-E-Commerce-Website-UI-UX-Design-(Community)?node-id=1-3&p=f&t=0ql7gTYOQNkxHj23-0

Refer to the cheat sheet here: https://www.sanity.io/docs/query-cheat-sheet

---

3. Shipment and Tracking Platform

Introduction to ShipEngine:

ShipEngine is a powerful shipping API platform designed to streamline the integration of shipping and tracking functionalities into your application. For the Hackathon, you must understand how this platform works to implement essential features like real-time shipment tracking and label generation.

Key Features of ShipEngine:

1. **Shipment Label Creation:**
   - Generate and print shipping labels for various carriers like FedEx, UPS, and USPS.
   - Customize labels with branding options.
2. **Tracking Shipments:**
   - Access real-time tracking updates for shipments.
   - Integrate tracking data into your frontend to display shipment progress.
3. **Rate Comparison:**
   - Compare shipping rates from different carriers to choose the most cost-effective option.
4. **Multi-Carrier Support:**
   - ShipEngine supports multiple carriers, enabling flexibility for businesses.

Steps to Integrate ShipEngine:

1. **Sign Up for ShipEngine:**
   - Create an account at ShipEngine.

2. **Get Your API Key:**

   - Navigate to the ShipEngine dashboard > API Settings.
   - Generate an API key for integration.

3. **Integrate ShipEngine with Your Application:**

Use HTTP requests to interact with ShipEngine's API endpoints.

Example:

```
const axios = require('axios');
const response = await axios.post('https://api.shipengine.com/v1/labels', {
 carrier_id: '<carrier_id>',
 service_code: 'priority',
 ship_to: {
  name: 'John Doe',
  address_line1: '123 Main St',
  city_locality: 'Austin',
  state_province: 'TX',
  postal_code: '78701',
  country_code: 'US'
 },
 ship_from: {
  name: 'Sender Name',
  address_line1: '456 Elm St',
  city_locality: 'Dallas',
  state_province: 'TX',
  postal_code: '75201',
  country_code: 'US'
 },
 packages: [
  {
   weight: {
    value: 1.5,
    unit: 'pound'
   }
  }
 ]
}, {
 headers: {
  'Authorization': 'Bearer <your_api_key>',
  'Content-Type': 'application/json'
 }
});

console.log(response.data);
```

4. **Testing the Integration:**
    - Use tools like Postman to test API requests before integrating them into your application.
    - Verify that shipment details and tracking updates are displayed correctly.

Practical Use in Hackathon:

- You will not implement ShipEngine during the Practice Hackathon, but understanding its functionality is critical for the actual Hackathon.
- Prepare mock shipment and tracking data to simulate real-world scenarios.

FAQs

1. **Can I modify my previous UI/UX template?**
    - Yes, you are encouraged to update, add, or remove pages as needed.
2. **What if my API data does not match the requirements?**
    - Ensure you test and align your mock API data structure with the schema provided in this guide.
3. **Will my UI/UX design be evaluated?**
    - No, the focus is solely on frontend functionality and integration with Sanity CMS.
4. **What should I do if I face issues with Sanity setup?**
    - Refer to the Sanity Documentation. https://www.sanity.io/docs
5. **Do I need to create shipment-related functionality in the Practice Hackathon?**
    - No, shipment functionality will only be part of the Hackathon. However, understanding it now will give you an advantage.
6. **Can I use external libraries for frontend development?**
    - Yes, you are free to use external libraries as long as they support your project requirements.
7. **How can I ensure my API schema matches the required data?**
    - Use the examples and guidelines provided in the Fake JSON API section and validate your data with tools like Postman.
8. **What tools are mandatory for this Hackathon?**
    - Tools like Sanity CMS, Next.js, and a code editor (e.g., VS Code) are mandatory.

9. **Can we collaborate with other teams?**
    - No, each candidate is required to work independently.
10. **How do I fetch images from the API and display them?**
    - Use the image URL provided in the API response and integrate it into your frontend code.
11. **What if I need more fields in my schema?**
    - Extend the schema as needed, ensuring compatibility with the overall project structure.
12. **How do I debug issues with Sanity integration?**
    - Use tools like the Sanity CLI and browser developer tools to inspect API calls and logs.
13. **What is the deadline for the Practice Hackathon?**
    - The deadline is strictly 12 January 2025.
14. **Can I switch from FakeStoreAPI to my own API?**
    - Yes, but ensure it meets the data requirements and matches your schema.
15. **Do we need to handle authentication in the Practice Hackathon?**
    - No, focus on CRUD operations and functionality, use customer cell no to register order.
16. **What if I encounter CORS issues while fetching data?**
    - Use a CORS proxy or configure headers appropriately in your API settings.
17. **Can I use a different UI framework besides Next.js?**
    - No, Next.js is mandatory as per the guidelines.
18. **Will there be penalties for missing the deadline?**
    - No penalties. It will affect in your Hackathon preparation.
19. **Can I pre-fill some data in Sanity for testing?**
    - Yes, pre-fill data to test your application, but ensure dynamic operations work correctly.
20. **How do I deploy my Next.js project for the Hackathon?**
    - Use platforms like Vercel or Netlify for deployment.