Mock-2 Practice Assessment - React Front-End Development

Total Marks: 60
Duration: 120 Minutes
Mode: Individual Assessment
Submission Format: ZIP File and PDF Report

Objective

This practice assessment evaluates your ability to design, build, and integrate React-based front-end applications covering key React concepts such as JSX, Components, Routing, State Management, Hooks, Context API, and PWA capabilities. You will be assessed on implementation quality, reusability, and adherence to component-based architecture.

Assessment Overview

User Story	Focus Area	Complexity	Marks
User Story 1	React Basics & Component Architecture	Easy	20
User Story 2	Routing, Data Integration & Lifecycle	Medium	20
User Story 3	Advanced Patterns, Hooks, and State Management	Difficult	20
Total			**60**

Epic 1 – Travel Booking Platform (React Front-End Development)

User Story 1 – Homepage & Destination Showcase (React Basics)

Scenario:
Develop a responsive **Travel Booking Homepage** that displays featured destinations and offers using React Components. The page should demonstrate usage of **JSX, Virtual DOM, Props, State**, and **Event Handling**.

Requirements

Criteria	Description	Marks
Project Setup	Initialize React app using create-react-app	3
Component Design	Create and render multiple functional components (Header, DestinationCard, Footer)	4
JSX & Props Usage	Pass props dynamically to components	4
State & Event Handling	Implement simple click events (e.g., “Add to Wishlist”) using React State	4
Styling	Apply Bootstrap classes for layout and responsiveness	5
Total Marks (User Story 1)		**20**

User Story 2 – Travel Package Management (Routing & Data Integration)

Scenario:
Implement a **Travel Package Management** module to display package details fetched from a backend (using `json-server`). Use **React Router** for navigation and **Component Lifecycle Methods** for data loading.

Requirements

Criteria	Description	Marks
Routing Setup	Configure routes (`/home`, `/packages`, `/contact`) using `react-router-dom`	4
Data Fetching	Fetch travel package data from `json-server` using Fetch API or Axios	4
Lifecycle Methods	Use `useEffect()` or Class Lifecycle methods for data initialization	4
Transition Effects	Apply route transition animation for navigation between pages	3
Data Presentation	Display packages dynamically in composable components	3
PropTypes Validation	Validate props for key components	2
Total Marks (User Story 2)		**20**

User Story 3 – Booking Form & Advanced State Handling (Formik, Hooks, Redux)

Scenario:
Build a **Booking Form Module** for users to book travel packages. Implement **Form Handling** using Formik and Yup, manage state using **React Context API** or **Redux**, and handle validation errors gracefully.

Requirements

Criteria	Description	Marks
Form Handling	Use Formik for controlled form inputs and Yup for validation	4
Context API / Redux	Manage global state for user booking data	5
Custom Hooks	Implement a custom hook for form submission logic	3
Error Boundary	Implement Error Boundary to handle component-level errors	3
PWA Support	Convert the application into a Progressive Web App	3
Code Readability	Maintain modular, reusable structure with clear comments	2
Total Marks (User Story 3)		**20**

Bonus Section (Optional – 10 Marks)

State Management & Data Synchronization (Redux or Flux Simulation)

Scenario:
Simulate a **Booking Status Dashboard** using **Flux or Redux architecture**. Implement an action dispatcher to update booking status and display the updated state in UI components.

Criteria	Description	Marks
Store Configuration	Create Store, Actions, and Reducers	3
Dispatcher / Middleware	Implement dispatcher logic for state flow	3
UI Integration	Display and update state changes dynamically	2
Code Structure	Maintain clear architecture and folder organization	2
Total Marks (Bonus)		**10**

Self-Evaluation Table

Skill Area	Confidence Level (Low / Medium / High)	Comments
JSX and Virtual DOM		

React Components (Functional/Class)		
Props, State, Event Handling		
Lifecycle Methods / Hooks		
React Router & Transitions		
Formik & Yup		
Context API / Redux / Flux		
PWA Configuration		

Submission Instructions

Format Options

Submit your work as a ZIP file containing:

- React project folder with all components
- Screenshots of output pages (Home, Packages, Booking Form)
- JSON server data file (if used)
- PWA manifest and service worker files

And

Submit a PDF report containing:

- Description of each user story
- Key code snippets
- Screenshots of final output
- 2-3 line explanation of your approach

Naming Convention

ParticipantName_MockPractice_React.zip

Example: ParthShah_MockPractice_React.zip

Folder Structure Example

```
/MockPracticeAssessment/
|-- userstory1/
|     ├── App.js
|     ├── DestinationCard.js
|     ├── style.css
|-- userstory2/
|     ├── PackageList.js
|     ├── routerConfig.js
|-- userstory3/
|     ├── BookingForm.js
|     ├── store/
|     |   ├── actions.js
|     |   ├── reducers.js
|-- screenshots/
```

```
|      ├── homepage.png
|      ├── packages.png
|      ├── bookingform.png
```

Execution Requirements

- All components must compile without error (`npm start`)
- Data fetched correctly via API or `json-server`
- PWA should run offline using service worker
- Include a `README.txt` for setup or dependencies

Deadline

Submit all files within 120 minutes.
Late submissions will result in a 10% deduction for every 15-minute delay.