EXP 2: Comparative Analysis of Naïve Prompting versus Basic Prompting Using ChatGPT Across Various Test Scenarios

Aim:

- To test how ChatGPT responds to naïve prompts (broad or unstructured) versus basic prompts (clearer and more refined) across multiple scenarios, analyzing the quality, accuracy, and depth of the generated responses.

Instructions:

1. **Define the Two Prompt Types:**
    - **Naïve Prompts:** Broad, vague, or open-ended prompts with little specificity.
    - **Basic Prompts:** Clear, detailed, and structured prompts that give specific instructions or context to guide the model.
2. **Prepare Multiple Test Scenarios:**
    - Select various scenarios such as:
        - Generating a creative story.
        - Answering a factual question.
        - Summarizing an article or concept.
        - Providing advice or recommendations.
        - Or Any other test scenario
    - For each scenario, create both a naïve and a basic prompt. Ensure each pair of prompts targets the same task but with different levels of structure.
3. **Run Experiments with ChatGPT:**
    - Input the naïve prompt for each scenario and record the generated response.
    - Then input the corresponding basic prompt and capture that response.
    - Repeat this process for all selected scenarios to gather a full set of results.
4. **Evaluate Responses :**

    Compare how ChatGPT performs when given naïve versus basic prompts and analyze the output based on Quality,Accuracy and Depth. Also analyse does ChatGPT consistently provide better results with basic prompts? Are there scenarios where naïve prompts work equally well?

Deliverables:

- A table comparing ChatGPT's responses to naïve and basic prompts across all scenarios.
- Analysis of how prompt clarity impacts the quality, accuracy, and depth of ChatGPT's outputs.
- Summary of findings with insights on how to structure prompts for optimal results when using ChatGPT.

Prompt :
give me a 30 days daily schedule of spending 2 hours per day to learn React JS with its concept names clearly .

Answer:

Here's a 30-day ReactJS learning schedule, with 2 hours per day dedicated to different concepts:

Week 1: React Basics

Day 1:

- Introduction to React
  - What is React?
  - Setting up the environment (Node, npm, create-react-app).
  - Overview of JSX and Virtual DOM.
  - First React app: "Hello World."

Day 2:

- JSX and Rendering Elements
  - Understanding JSX syntax.
  - Rendering HTML elements in React.
  - Embedding expressions in JSX.

Day 3:

- Components (Functional vs Class)
  - Introduction to components.
  - Creating and rendering functional components.
  - Basic structure of class components.

Day 4:

- Props in React

  - Passing data between components using props.
  - Understanding how props are immutable.

Day 5:

- State in React (Class Components)
  - Introduction to state in class components.
  - Setting and updating state.

Day 6:

- Handling Events in React
  - Handling events (onClick, onChange, etc.).
  - Passing methods as props to child components.

Day 7:

- Project Day
  - Build a small project: Create a basic "Todo List" app.
  - Use state and props to manage and display the tasks.

Week 2: Advanced React Concepts

Day 8:

- Lifecycle Methods (Class Components)
  - Introduction to React lifecycle.
  - Methods: componentDidMount, componentDidUpdate, componentWillUnmount.

Day 9:

- Conditional Rendering
  - Using conditional operators (if, ternary, &&) to render components dynamically.

Day 10:

- Lists and Keys
  - Rendering lists in React using map().
  - Understanding the importance of keys in React lists.

Day 11:

- Forms in React
    - Handling form inputs.
    - Managing form submission and capturing form data.

Day 12:

- Controlled vs Uncontrolled Components
    - Difference between controlled and uncontrolled form inputs.
    - Working with uncontrolled components using refs.

Day 13:

- Project Day
    - Enhance the "Todo List" app:
        - Add functionality to edit and delete tasks.
        - Use forms to add new tasks.

Day 14:

- Hooks Introduction (useState)
    - Introduction to Hooks in functional components.
    - Replacing class-based state with useState.

Week 3: React Hooks and Context API

Day 15:

- useEffect Hook
    - Handling side effects in functional components.
    - Replacing lifecycle methods with useEffect.

Day 16:

- More Hooks (useRef and useReducer)
    - Using useRef for DOM manipulation and preserving values.
    - Introduction to useReducer for complex state management.

Day 17:

- Custom Hooks
    - Creating and using custom hooks for code reuse.

Day 18:

- Context API
    - Introduction to Context API.
    - Using Context to manage global state without props drilling.

Day 19:

- useContext Hook
    - Accessing and updating global state using useContext.

Day 20:

- React Router (Part 1)
    - Introduction to client-side routing.
    - Installing and setting up react-router-dom.
    - Basic route configuration.

Day 21:

- React Router (Part 2)
    - Dynamic routing and route parameters.
    - Using Link, NavLink, and Redirect.

Week 4: Advanced Concepts and Project

Day 22:

- Higher-Order Components (HOCs)
    - Understanding the concept of HOCs.
    - Writing reusable components using HOCs.

Day 23:

- React Portals
    - Introduction to React Portals for rendering components outside the main DOM hierarchy.

Day 24:

- Error Boundaries
    - Catching JavaScript errors in React components using Error Boundaries.

Day 25:

- React Fragments and Strict Mode
    - Using fragments to avoid unnecessary DOM elements.
    - Enforcing best practices with Strict Mode.

Day 26:

- React Performance Optimization
    - Optimizing rendering with React.memo, useMemo, and useCallback.
    - Lazy loading components with React.lazy() and Suspense.

Day 27:

- Testing in React
    - Introduction to testing in React using Jest and React Testing Library.
    - Writing basic unit tests for components.

Day 28:

- Project Day
    - Build a React project: Create a "Weather App" using an API.
    - Practice using useEffect for fetching data and useState for managing it.

Day 29:

- Deploying React Applications
    - Preparing your React app for production.
    - Deploying the app on Netlify or GitHub Pages.

Day 30:

- Final Project
    - Build a React app of your choice (e.g., a portfolio site or blog).

  - Implement everything learned: routing, state management, hooks, and API integration.

Prompt:

suggest me calories deficient diet foods, meal that based on south indian where it should be within 1500 calories per day.

ChatGPT reponse:

Here's a 1500-calorie South Indian meal plan that incorporates common foods, providing a balance of nutrients while helping you maintain a calorie-deficit diet:

Breakfast (300-350 calories)

Option 1:

- Idli (2 medium-sized): ~120 calories
- Sambar (1 cup): ~150 calories
- Coconut Chutney (2 tbsp): ~50 calories

Option 2:

- Upma (1 cup): ~250 calories
- Green Chutney (2 tbsp): ~30 calories
- Black Coffee/Green Tea: 0 calories

Option 3:

- Dosa (1 medium-sized): ~150 calories
- Tomato Chutney (2 tbsp): ~30 calories
- Boiled Egg (1): ~70 calories
- Filter Coffee (without sugar): ~50 calories

Mid-Morning Snack (100-150 calories)

Option 1:

- 1 Banana: ~90 calories

- Buttermilk (1 glass): ~50 calories

Option 2:

- Roasted Peanuts (1 handful): ~100 calories

Option 3:

- Mixed Fruits (1 small bowl - Papaya/Apple): ~100 calories

Lunch (400-450 calories)

Option 1:

- Brown Rice (1 cup cooked): ~215 calories
- Sambar (1 cup): ~150 calories
- Cucumber Salad (1 small bowl): ~30 calories
- Curd (1/2 cup): ~60 calories

Option 2:

- Chapati (2 medium-sized): ~160 calories
- Vegetable Kurma (1 cup): ~150 calories
- Curd (1/2 cup): ~60 calories
- Stir-fried Vegetables (1/2 cup): ~70 calories

Option 3:

- Quinoa (1 cup cooked): ~200 calories
- Rasam (1 cup): ~50 calories
- Paneer Bhurji (1/2 cup): ~150 calories
- Kachumber Salad (1 small bowl): ~40 calories

Evening Snack (100-150 calories)

Option 1:

- Masala Buttermilk (1 glass): ~60 calories
- Roasted Chickpeas (1 handful): ~90 calories

Option 2:

- 1 small bowl of Sprouts Salad (with onion, tomato, lemon): ~120 calories

Option 3:

- Coconut Water: ~40 calories
- 1 boiled egg: ~70 calories

Dinner (400-450 calories)

Option 1:

- Ragi Dosa (2 medium-sized): ~200 calories
- Tomato Chutney (2 tbsp): ~30 calories
- Vegetable Curry (1 cup): ~180 calories

Option 2:

- Vegetable Soup (1 bowl): ~100 calories
- Paneer Tikka (1 cup): ~150 calories
- Chapati (1 medium-sized): ~80 calories

Option 3:

- Grilled Fish (100g): ~200 calories
- Mixed Veg Stir Fry (1 small bowl): ~150 calories
- Curd (1/2 cup): ~60 calories

Before Bed (Optional Snack - 50 calories)

- Warm Milk (1/2 cup): ~50 calories

This plan focuses on traditional South Indian foods while ensuring that the meals are nutrient-dense and under 1500 calories per day. You can mix and match the options to create a variety of meals throughout the week.