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.

Procedure:

CASE 1 : Creating a customized roadmap

NAIVE PROMPT :

You said:
Provide a roadmap for front end development

OUTPUT:

ChatGPT said:
ChatGPT

Here's a structured **roadmap for Front-End Development** that can guide you through the essential concepts and technologies, broken down into beginner, intermediate, and advanced stages:

1. Beginner Stage

a. HTML (HyperText Markup Language)

- Learn the basic structure of a webpage (head, body, and elements).
- Master HTML elements: headings, paragraphs, links, images, lists, forms, tables, and semantic HTML.
- Understand how to create accessible and SEO-friendly web pages.

b. CSS (Cascading Style Sheets)

- CSS Syntax and Selectors (class, ID, element selectors).
- Layouts: Box Model, Flexbox, Grid Layout.
- Responsive Web Design: Media queries, fluid grids, and percentages.
- Basic animations and transitions.

c. JavaScript (JS)

- JavaScript syntax: variables, data types, operators.
- DOM Manipulation: Selecting, creating, and modifying elements.
- Events and Event Listeners (click, hover, input, etc.).
- Functions and ES6+ features: arrow functions, let/const, template literals.

d. Version Control (Git & GitHub)

- Learn the basics of Git (commit, push, pull, branch).
- Collaborate using GitHub: repositories, pull requests, merging, and Git workflows.

---

2. Intermediate Stage

a. Advanced JavaScript

- Asynchronous JavaScript: Promises, async/await, Fetch API, and handling API requests.
- Object-Oriented Programming (OOP) in JavaScript: Classes, constructors, inheritance.
- Modules: ES6 modules, import/export.

b. CSS Preprocessors

- Learn about **Sass** or **LESS** to write more efficient CSS.
- Variables, mixins, and nesting in preprocessors.

c. Web Development Tools

- **Package Managers**: NPM/Yarn to manage project dependencies.
- **Build Tools**: Webpack, Parcel, Vite to bundle and optimize resources.
- **Task Runners**: Gulp/Grunt for automating repetitive tasks like minification, transpiling, etc.

d. Responsive Design & CSS Frameworks

- Master Flexbox and Grid systems for complex layouts.
- Use frameworks like **Bootstrap** or **TailwindCSS** to accelerate UI development.
- Dive deeper into media queries and cross-browser compatibility.

e. JavaScript Frameworks/Libraries

- **React.js**: Components, JSX, Props, State, and Lifecycle Methods.
- Basic routing with **React Router**.
- Introduction to state management with **Redux** or **Context API**.

---

3. Advanced Stage

a. Advanced JavaScript Concepts

- **JavaScript Design Patterns** (Factory, Singleton, Observer, etc.).
- Performance optimization techniques: Debouncing, Throttling, Lazy loading.
- **Error Handling**: Try-catch, debugging tools (Chrome DevTools, etc.).

b. Advanced React

- Dive deeper into Hooks: useEffect, useContext, custom hooks.
- Component patterns: Higher-Order Components (HOCs), Render Props.
- Performance optimization: Memoization (React.memo, useCallback), code splitting.

c. State Management

- Advanced **Redux** (Thunks, Middleware, Selectors, and reselect).
- Using **React Query** for server-state synchronization.

d. Testing

- **Unit Testing**: Using **Jest** or **Mocha** to test JavaScript logic.
- **Component Testing**: Test React components using **React Testing Library** or **Enzyme**.
- End-to-end testing with **Cypress** or **Puppeteer**.

e. TypeScript

- Introduction to TypeScript: Types, Interfaces, Generics.
- Benefits of using TypeScript with React (strong typing, error reduction).

f. Progressive Web Apps (PWAs)

- Learn to build PWAs: Service workers, web manifests, caching strategies, and offline capabilities.
- Add-to-home-screen functionality, push notifications.

g. Web Performance

- **Lighthouse**: Use to audit web performance.
- Learn about **Web Vitals** (FCP, LCP, TTI, etc.).
- Implement lazy loading, code splitting, image optimizations.

h. Web Security

- Learn about web security best practices: Content Security Policy (CSP), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and HTTPS.

Additional Skills

- **Version Control (Advanced)**: Branching strategies, pull requests, and working with teams.
- **UI/UX Design Basics**: Understanding the principles of design, typography, color theory, and using design tools like Figma or Adobe XD.
- **APIs & Authentication**: Understand how to work with REST APIs, GraphQL, and integrate authentication using OAuth or JWT.
- **Deploying Applications**: Learn how to deploy applications using services like **Netlify**, **Vercel**, or **GitHub Pages**.

Practice

- Build multiple projects along the way (e.g., portfolios, to-do apps, blogs, e-commerce sites, etc.).
- Contribute to open-source projects to gain real-world experience.

This roadmap will guide you in becoming proficient in front-end development, and you can adjust it according to your pace and learning style.

REFINED PROMPT(INTERVIEW PATTERN):

You said:
From now onwards ask question one by one and customize a roadmap for front end development

ChatGPT said:
ChatGPT

Memory updated

Let's start by understanding your current level.

Do you have any prior experience with front-end development, or are you starting from scratch?

You said:
I have an intermediate level of experience

ChatGPT said:
ChatGPT

Memory updated

Great! Since you're at an intermediate level, we can focus on enhancing your existing skills and moving toward advanced topics.

To customize this roadmap, what is your main goal in front-end development? For example, do you want to improve your proficiency with specific frameworks (like React), work on performance optimization, or perhaps explore more modern tools and design techniques?

You said:
As I am proficient in React,I wanted to explore the next stages of development process

ChatGPT said:
ChatGPT

Memory updated

Since you're proficient in React, let's focus on taking your skills to the next level. We can dive deeper into performance optimization, advanced state management, testing, and other modern front-end practices.

To start, have you already worked with React hooks like useEffect, useContext, or custom hooks?

You said:
Yes.To some extent I have worked with React hooks

ChatGPT said:
ChatGPT

Memory updated

Great! Since you've worked with React hooks, the next step could be to deepen your understanding of custom hooks and their use cases for reusability and optimization.

Would you like to dive deeper into **performance optimization** in React (e.g., memoization, lazy loading, code splitting), or explore advanced topics like **state management** with tools like Redux or React Query?

You said:
Performance Optimization

ChatGPT said:
ChatGPT

To explore **performance optimization** in React, here are some key concepts and techniques we can cover:

1. **Memoization** with React.memo, useMemo, and useCallback to prevent unnecessary re-renders.
2. **Code Splitting** to load components and libraries only when needed, using tools like React's Suspense and dynamic import().
3. **Lazy Loading** for images and components to improve initial load times.
4. **React Profiler** for measuring performance and identifying bottlenecks.
5. Optimizing **list rendering** using React.Fragment, keys, and virtualization (e.g., react-window or react-virtualized).

Would you like to start with **memoization techniques** or jump into **code splitting and lazy loading**?

You said:
We can start with memoization techniques

ChatGPT said:
ChatGPT

Memoization is a powerful optimization technique in React that helps avoid unnecessary re-renders and recalculations. Let's break it down into three key areas:

1. React.memo()

- React.memo is a higher-order component (HOC) that prevents a functional component from re-rendering if its props haven't changed.

  Example:

  javascript
  Copy code
  ```
  const ChildComponent = React.memo(({ data }) => {
    console.log('Child rendered');
    return <div>{data}</div>;
  });
  ```

- If data remains the same between renders, ChildComponent won't re-render, improving performance.

2. useMemo()

- useMemo caches the result of a computation and only recalculates it when its dependencies change.

  Example:

javascript
Copy code
```
const expensiveCalculation = useMemo(() => {
  return calculateSomething(value);
}, [value]);
```

- This is useful for expensive operations, as React will only recalculate when value changes.

3. useCallback()

- useCallback is used to memoize callback functions. It prevents the creation of new function instances on every render, which can cause unnecessary re-renders of child components.

  Example:

  javascript
  Copy code
  ```
  const handleClick = useCallback(() => {
    doSomething();
  }, [dependency]);
  ```

- If dependency doesn't change, handleClick will retain the same reference between renders, helping with performance.

To start optimizing your React code:

- Use React.memo for functional components that receive props but don't need to re-render unless those props change.
- Apply useMemo for expensive calculations to avoid recalculating them on every render.
- Utilize useCallback when passing functions as props to prevent unnecessary re-renders of child components.

COMPARISON:

In the **naive pattern**, the prompt directly asks for a complete front-end roadmap, resulting in a generalized, static response without much interaction. On the other hand, the **interview pattern** takes an iterative approach, gathering more context about your skills and goals, leading to a tailored, step-by-step roadmap that adapts as the conversation progresses. The interview pattern allows for deeper exploration and customization based on your input.

CASE 2: RESUME CHECKER

NAIVE PROMPT:

You said:
Resume-3.pdf

Analyze this resume

OUTPUT:

ChatGPT said:
ChatGPT

The resume you uploaded presents a dynamic profile of Rehan Naveid R, a second-year Electronics and Communication Engineering (ECE) student. Here's an analysis:

Strengths:

1. **Clear Objective**: The resume communicates a clear objective of seeking internship opportunities to leverage coding skills, with a focus on software development and web-based projects.
2. **Education and Performance**: The CGPA of 8.5 in B.E. and high academic scores in previous education levels (91.6% in SSLC and 89% in HSC) demonstrate strong academic performance.
3. **Relevant Skills**: Rehan has proficiency in Python and C programming languages, as well as experience in machine learning and web development technologies. This includes frameworks like OpenCV, Scikit-learn, TensorFlow, and Mediapipe, and skills in frontend development with HTML, CSS, and JavaScript.
4. **Project Experience**: Rehan's project on "Dynamic Gesture Recognition System" shows hands-on experience with machine learning and computer vision, demonstrating a practical application of skills.
5. **Certifications**: The certifications listed, such as "Introduction to Back end development" by Meta and NPTEL's course on "Understanding Incubation and Entrepreneurship," add value by indicating continuous learning and interest in software and entrepreneurial skills.
6. **Soft Skills**: Mention of soft skills like team management, adaptability, and problem-solving is positive for potential employers.
7. **LinkedIn and GitHub Presence**: Including links to LinkedIn and GitHub profiles is beneficial, providing potential employers with easy access to Rehan's professional network and project work.

Areas of Improvement:

1. **More Specific Details on Projects**: While the project is mentioned, adding specific details such as the programming languages and frameworks used, the challenges faced, and the results or outcomes would make it more impactful.
2. **Experience Section**: The resume does not currently list any internships, part-time roles, or extracurricular activities that show experience outside of academics. Including such information would help demonstrate practical application of skills.
3. **Refinement in Format**: Some parts of the resume feel a bit cluttered, such as the skills section. Breaking it down into categories like "Programming Languages," "Frameworks," "Tools," etc., would enhance readability.
4. **Soft Skills Elaboration**: Instead of just listing soft skills, Rehan could provide examples where these skills were applied in projects or teamwork situations to show evidence of them in action.

Overall Assessment:

Rehan presents a strong foundation in academics and technical skills, but could further enhance the resume by expanding on practical experience, refining the format, and providing more detail on how skills have been applied. This will help create a stronger impression for internship opportunities.

REFINED PROMPT(PERSONA PATTERN):

You said:
Resume-3.pdf

PDF

From now on ,as an ATS resume checker Analyze this resume and mention the areas of improvements

2/2

OUTPUT:

ChatGPT said:

Analyzing the resume from an Applicant Tracking System (ATS) perspective, here are some observations and areas for improvement:

ATS-Friendliness:

1. **Consistent Formatting and Section Headers**:
   - **Improvement Needed**: Ensure that section headers are clear and consistently formatted to help ATS software parse the information correctly. For example, the "PROJECT" and "CERTIFICATIONS" sections lack standard formatting, which may confuse ATS systems.
   - **Recommendation**: Standardize headers by using consistent fonts, sizes, and bolding across all sections (e.g., "Education," "Skills," "Certifications," "Projects"). Include more traditional section labels like "Experience" or "Work History" even if the experience is project-based.
2. **Skills Section Optimization**:
   - **Improvement Needed**: While the skills are listed, they are somewhat cluttered and not organized into clear categories. ATS software might struggle to extract all the relevant keywords.
   - **Recommendation**: Break down the skills section into clearer categories such as:
     - **Programming Languages**: Python, C
     - **Frameworks and Libraries**: Django, OpenCV, TensorFlow, Scikit-learn, Mediapipe
     - **Web Development**: HTML, CSS, JavaScript
     - **Databases**: SQL
     - **Soft Skills**: Problem-solving, Team management, Adaptability This structured approach helps ATS software identify the relevant skills quickly.
3. **Keyword Density**:
   - **Improvement Needed**: ATS systems scan for specific keywords relevant to the job description. The resume lists key programming languages and skills but lacks depth in describing where and how these skills were used.
   - **Recommendation**: Add more context around each skill within project descriptions or experience sections. For example, mention how Python was used in machine learning projects or how JavaScript was applied in web development. Including keywords like "frontend development," "backend development," or specific frameworks such as "React" or "Node.js" could enhance ATS visibility.
4. **Project Details**:
   - **Improvement Needed**: While the "Dynamic Gesture Recognition System" project is mentioned, it does not include enough quantifiable outcomes or technical details that ATS systems look for.
   - **Recommendation**: Provide measurable results and more specific descriptions, such as the percentage of accuracy in the sign language recognition system, data processed, or the technology stack used. For example: "Built a real-time gesture recognition system using Python and TensorFlow, achieving 95% accuracy in recognizing sign language gestures."
5. **Certifications**:

    - **Improvement Needed**: The certifications section is somewhat buried within the resume.
    - **Recommendation**: Create a distinct "Certifications" section with clear formatting. This will make it easier for ATS to pick up these credentials, which are often important for certain roles.
6. **Contact Information Placement**:
    - **Improvement Needed**: The contact information is mixed with other details, which could confuse some ATS systems.
    - **Recommendation**: Place contact information (email, phone number, LinkedIn, GitHub) at the top of the resume, separated from other sections for easier parsing.
7. **Avoid Cluttering Keywords**:
    - **Improvement Needed**: The skills section has overlapping keywords (e.g., "OpenCV, Scikit-learn, Django, TensorFlow, MediapipeLibraries/Frameworks"). This may confuse ATS software.
    - **Recommendation**: Separate each skill with commas and ensure there is space between words. For example: "Libraries/Frameworks: OpenCV, Scikit-learn, Django, TensorFlow, Mediapipe."

General Improvements for ATS:

- **Action-Oriented Descriptions**: Each project or experience entry should begin with strong action verbs (e.g., "Developed," "Led," "Designed") and be followed by specific results. This not only aids readability but also ensures ATS systems recognize your achievements.
- **Tailor the Resume**: Depending on the job you're applying for, tailor the resume to match the job description closely by emphasizing the skills and experiences that are relevant to that specific role.

COMPARISON:

In this persona-based prompting pattern, the AI is asked to **adopt a role** (ATS resume checker), which provides a clear lens through which to perform the task. This helps ensure the response is more focused and task-oriented, optimizing the resume for ATS rather than offering generic feedback.

CASE 3: GENERATING CAMPAIGNING IDEAS

NAIVE PROMPT:

You said:
Give me marketing ideas for an app

OUTPUT:

ChatGPT said:
ChatGPT

Here are several marketing ideas to promote your fitness app targeting busy professionals:

1. **Content Marketing**:
    - Create a blog or vlog series focusing on fitness tips, healthy meal prep, and quick workout routines tailored for busy schedules. Share success stories and testimonials from users.
2. **Social Media Campaigns**:
    - Utilize platforms like Instagram, LinkedIn, and Twitter to engage your audience. Share motivational quotes, fitness challenges, and user-

generated content. Consider hosting live workout sessions or Q&A segments.

3. **Influencer Partnerships**:
    - Collaborate with fitness influencers who resonate with your target audience. They can help promote your app through authentic endorsements, tutorials, and challenges.
4. **Email Marketing**:
    - Develop a newsletter that offers fitness tips, app updates, and exclusive promotions. Provide value by sharing resources like meal plans or workout schedules.
5. **Referral Program**:
    - Create a referral program where users can earn rewards for inviting friends to download and use the app. This encourages word-of-mouth marketing.
6. **Corporate Wellness Programs**:
    - Partner with companies to offer your app as part of their employee wellness programs. Provide a free trial or discount for employees to encourage sign-ups.
7. **Free Trials and Promotions**:
    - Offer a limited-time free trial to attract new users. Consider promotions around New Year's or other health-centric events.
8. **User Challenges**:
    - Launch monthly fitness challenges within the app. Encourage users to share their progress on social media using a specific hashtag to increase visibility.
9. **Targeted Advertising**:
    - Use paid advertising on social media platforms and Google Ads to reach busy professionals. Target ads based on interests and demographics related to fitness and health.
10. **Webinars and Workshops**:
    - Host online webinars or workshops on topics like "Fitness for Busy Professionals" or "Healthy Meal Prep." Use these platforms to showcase your app's features.
11. **Engagement through Gamification**:
    - Incorporate gamification elements in your app, such as badges, points, or leaderboards, to encourage user engagement and competition.
12. **Local Community Events**:
    - Participate in or sponsor local fitness events or fairs. Set up booths to showcase your app and offer demos or freebies.

Implementing a mix of these strategies can help you effectively reach and engage your target audience while building a strong brand presence in the competitive fitness app market.

REFINED PROMPT(FEW SHOT PROMPT):

Suggest marketing campaign ideas for a new fitness app targeting busy professionals. Example 1:"30-Day Challenge: Boost Your Energy with Quick Workouts!" Example 2:"Lunch Break Fitness: 15-Minute Routines for the Office!"

OUTPUT:

ChatGPT said:
ChatGPT Here are some marketing campaign ideas for your fitness app targeting busy professionals:

1. **"Fit in 10: Quick Workouts for Your Morning Rush!"**
    - Promote a series of 10-minute workouts designed for mornings, helping users kickstart their day without taking too much time.
2. **"Desk to 5K: Train on Your Lunch Break!"**
    - Create a training program for busy professionals aiming to run a 5K, featuring workouts they can do during lunch breaks.
3. **"Mindful Minutes: Stress Relief Techniques for Busy Minds!"**
    - Offer short mindfulness exercises or stretching routines that can be done in under 5 minutes to help professionals de-stress during the workday.
4. **"Weekend Warrior: 30-Minute Full-Body Workouts!"**
    - Encourage users to make the most of their weekends with efficient full-body workout routines that fit into busy schedules.
5. **"Snack-Sized Nutrition: Healthy Recipes in Under 10 Minutes!"**
    - Share quick and nutritious recipes that professionals can whip up in minutes, promoting the idea of healthy eating without the hassle.
6. **"Step Up: 5,000 Steps Challenge for the Office!"**
    - Launch a step challenge encouraging users to increase their daily steps, with tips on how to incorporate movement into their workday.
7. **"Workout Wednesdays: Midweek Motivation with Quick Challenges!"**
    - Introduce weekly fitness challenges every Wednesday to keep users engaged and motivated throughout the week.
8. **"After-Hours Fitness: Evening Workouts to Unwind!"**
    - Create a campaign around evening routines that help professionals unwind after a long day, focusing on low-impact workouts or yoga sessions.
9. **"Fitness in Transit: On-the-Go Workouts for Busy Professionals!"**
    - Develop a series of workouts that can be done in hotel rooms or during travel, appealing to professionals who travel for work.
10. **"Social Sweat Sessions: Group Challenges for Team Building!"**
    - Encourage corporate teams to engage in group fitness challenges using the app, promoting camaraderie and healthy competition.
11. **"Motivation Monday: Start Your Week with a Quick Boost!"**
    - Every Monday, share a quick workout or motivational message to inspire users to kick off their week actively.
12. **"Fitness Reminders: Your Personal Trainer in Your Pocket!"**
    - Promote a feature that allows users to set reminders for workouts, stretching, or hydration to help them stay on track throughout the day.

These campaigns can create engagement and drive downloads while addressing the specific needs and constraints of busy professionals.

COMPARISON:

A **naive prompt** like "Explain marketing strategies for a fitness app" is straightforward but might lead to vague or less relevant answers. On the other hand, a **few-shot prompt**, such as providing examples like "30-Day Challenge: Boost Your Energy with Quick Workouts!" and "Lunch Break Fitness: 15-Minute Routines for the Office!", gives clear guidance. This helps generate more specific and useful ideas, leading to better-quality responses overall.

OVERALL COMPARISON TABLE:

Criteria	Naïve Prompting	Basic Prompting (Interview/Persona-Based)
Approach	Direct, broad, and unstructured requests for a complete answer.	Step-by-step or role-specific approach, gathering context and adapting the response.
Depth of Response	Provides general or static information without much customization.	Tailored responses based on user input, leading to more in-depth exploration.
Interactivity	Minimal interaction, often a one-time, non-adaptive response.	High interactivity with questions or clarifications to personalize and enhance the response.
Customization	Generic and less personalized.	Highly customized based on user's level, goals, or specific request.
Flexibility	Less flexible, delivering predefined responses.	More flexible, adapting to user needs as more details are provided.
Engagement	Limited, as the response doesn't prompt further interaction.	Encourages continuous dialogue, prompting for more input and deeper discussion.
Relevance	Can miss specific user needs due to lack of contextual understanding.	Addresses specific user needs by gathering details during the interaction.
Scenario 1: Customized Roadmap	A detailed yet static roadmap for front-end development without assessing the user's level.	Iterative and customized based on the user's skill level and specific goals.
Scenario 2: Resume Analysis	General feedback on the resume's content and structure.	ATS-specific analysis with precise formatting suggestions for optimization.
Scenario 3: Campaigning Ideas	Provides broad marketing strategies for an app without target specification.	Marketing ideas tailored based on a specific user base (e.g., fitness for busy professionals).

RESULT :

Naive Prompting offers general, static responses with minimal customization, while **Basic Prompting** tailors responses through interactivity, gathering user-specific details for deeper insights.