Surendra Panpaliya
Founder and CEO, GKTCS Innovations
https://www.linkedin.com/in/surendrarp

Surendra Panpaliya: AI Visionary

Extensive Digital Transformation Experience

With over 25 years in IT, his expertise drives digital transformation and technological innovation for global organisations.

Empowering IT Professionals

He has mentored and trained more than 25,000 IT professionals, equipping them with advanced technology skills and knowledge.

Advancing AI Adoption

Through collaborations with numerous multinational firms, he has promoted the adoption of AI-driven strategies across diverse industries.

Agenda

- Introduction to Prompt Engineering
- Understanding Prompt Structure
- Hands-On: Crafting Effective Prompts
- Role-Based Demo – HR Team

Introduction to Prompt Engineering

What is Prompt Engineering?

Evolution with LLMs (GPT-3 → GPT-4 → GPT-5 & Multi-modal models).

Applications across HR, QA, Development, BA domains.

Prompt types: **Zero-shot, Few-shot, Chain-of-Thought**.

Understanding Prompt Structure

Key components of a prompt (Instruction, Context, Example, Constraint).

Format styles: questions, templates, structured tasks.

Prompting for **reasoning vs. factual accuracy**.

Hands-On: Crafting Effective Prompts

Activity: Basic vs. improved prompts.

Role prompting, style prompting, task chaining.

Tools: **ChatGPT, OpenAI Playground**.

Role-Based Demo – HR Team

Demo	Demo: Creating prompts for HR tasks.
Case	Case Study (HR 1): Automating job description drafting.
Case	Case Study (HR 2): Resume screening & shortlisting with ethical safeguards.

Prompt Engineering Introduction

Surendra Panpaliya
International Corporate Trainer
GKTCS Innovations

Prompt Engineering Introduction

Art and science of designing

Effective instructions (prompts)

To get accurate, reliable, and

useful responses from AI systems

like ChatGPT, Copilot, Gemini, Claude

Prompt Engineering Introduction

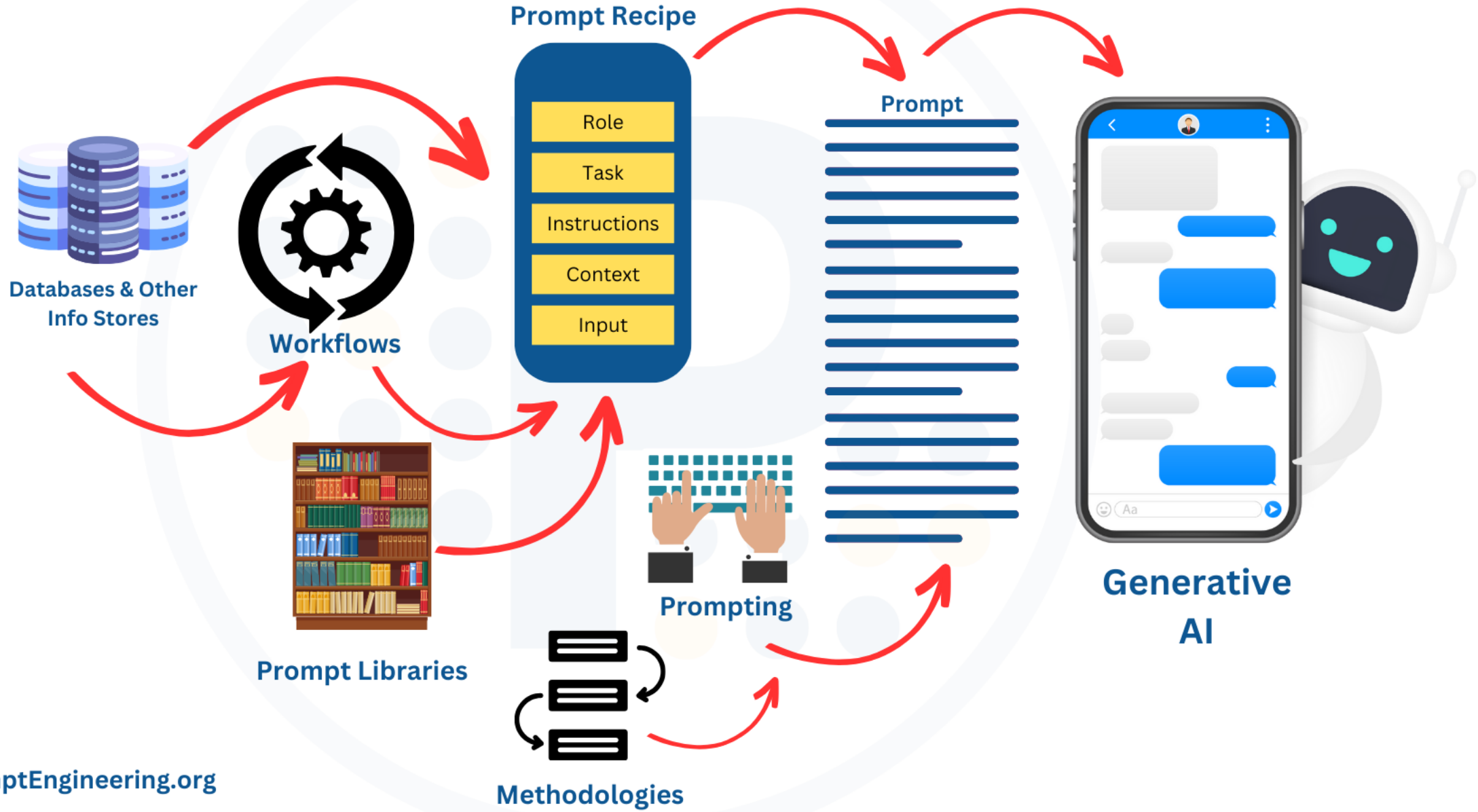
Think of it as

Asking the right question in the right way

So the AI understands your intent and

Produces relevant outputs.

What is Prompt Engineering? Everything that goes before the prompt



PromptEngineering.org

Prompt Example

You are an expert in prompt engineering for large language models like ChatGPT. Create a comprehensive, step-by-step guide for beginners on how to become proficient at prompt engineering. The guide should include clear explanations, practical examples, and common mistakes to avoid. Break down the process into logical phases (e.g., understanding the model, prompt structure, refining prompts, testing, etc.). Use a friendly and instructive tone, and format the guide clearly with headings, bullet points, and examples. Conclude with actionable tips and resources for further learning.

Why this prompt works?

Role assignment:

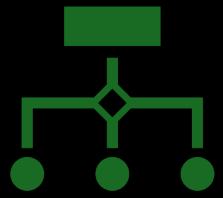
Tells the model

it's an expert

improves response quality.

Why this prompt works?

Clear goal:



“Create a step-by-step guide”

sets a direct objective.

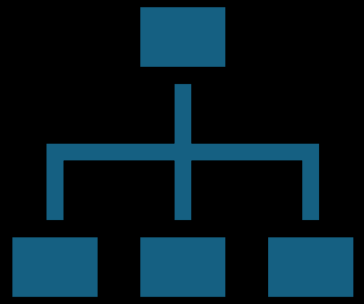
Why this prompt works?

Audience level:

“For beginners”

ensures explanations are accessible.

Why this prompt works?



Structure and formatting instructions:

Headings, bullets, examples — this guides clarity.

Why this prompt works?

TONE GUIDANCE:

“FRIENDLY AND INSTRUCTIVE”

KEEPS IT READABLE AND HELPFUL.

Why this prompt works?

Extras:

Asks for mistakes to avoid +

further resources

adds practical value.

A.C.T.O.R.S Framework

AIM

CONTEXT

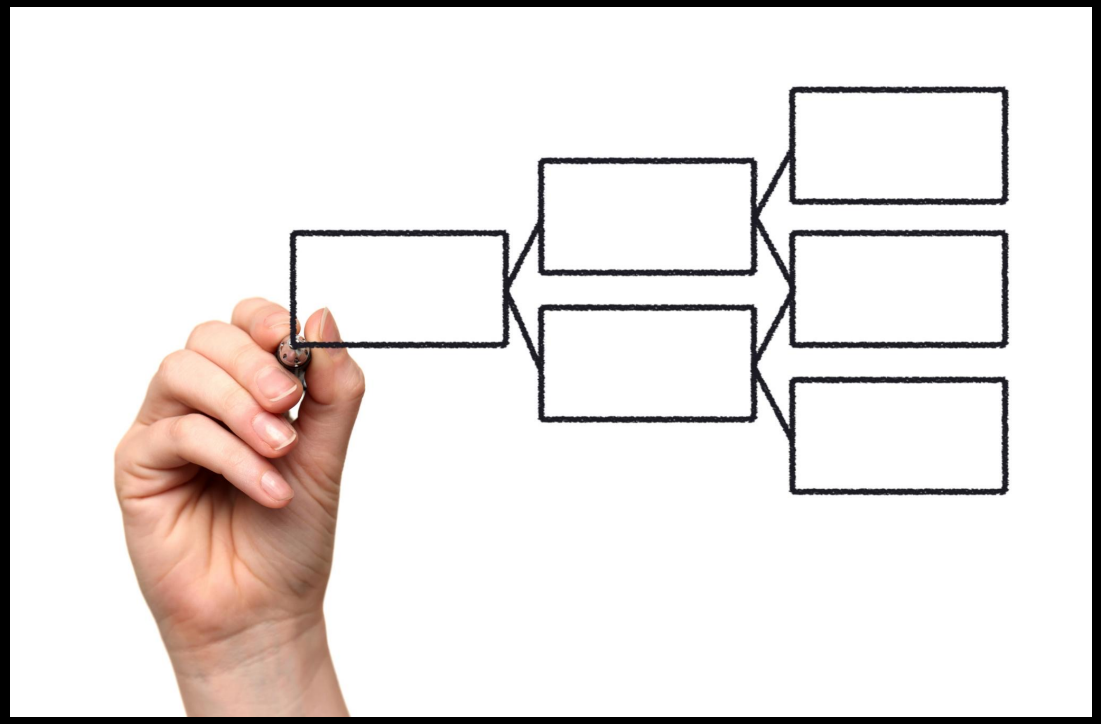
TASKS

OUTPUT FORMAT

RULES & CONSTRAINTS

SHOW EXAMPLES (FEW-SHOT)

A.C.T.O.R.S Framework Essentials



- Aim and Context define the purpose and background of the prompt.
- Tasks specify detailed actions the AI should perform.
- Output format determines how the result should be presented.
- Rules and constraints set boundaries and guidelines for responses.
- Examples provide clarity through sample inputs and desired outputs.

Prompt Structure (A.C.T.O.R.S.)

System (role): You are a helpful, rigorous {role}.
Follow constraints strictly.

User:
Aim: {what to achieve, success criteria}

Context: {domain, audience, source material, data snippets}

Prompt Example

System: You are an expert prompt engineer and teacher.

Be friendly, concise, and highly structured.

User:

Create a step-by-step beginner’s guide to prompt engineering that includes:

Prompt Structure (A.C.T.O.R.S.)

Tasks:

1) {subtask A}
2) {subtask B}
3) {subtask C}

Prompt Example

- Phases: understanding the model, goal-setting,

prompt structure, techniques (role/few-shot/decomposition/self-checks), refinement loop, testing/evals, common mistakes.

- Clear headings, bullets, and short examples for each phase.

- At least two copy-paste prompt templates (general + strict JSON).

- A mini QA checklist and a 7-day practice plan.

- Conclude with actionable tips and a short list of authoritative resources.

Prompt Structure (A.C.T.O.R.S.)

Output format: {JSON schema / Markdown sections / tables}

Rules & constraints:
- Tone: {e.g., friendly, concise}
- Limits: {word/char caps, IST dates, INR units}
- Sourcing: {what you may/may not assume}
- If unsure: say “I don’t know”.

Prompt Example

Constraints:

- Use Indian conventions if relevant (e.g., IST, INR).
- Keep explanations crisp; avoid filler.
- If you’re unsure about something, say “I don’t know.”
- No invented citations; list only well-known, official resources.

Prompt Structure (A.C.T.O.R.S.)

Examples:

Input → {short example}

Desired Output → {short, realistic example}

Prompt Example

Output format:

Title

Phase 1 ...

...

Resources

Why This Prompt Works

Expert Role Assigned

Giving the model an expert role improves the quality and depth of the generated answers, making them more insightful.

Clear Goal Set

Defining a clear objective, like creating a step-by-step guide, leads to more focused and actionable responses.

Audience Level Specified

Targeting explanations for beginners makes content more accessible and easy to understand for newcomers.

Mastering Prompt Engineering

Clarity and Structure

Prompts should be structured with clear instructions, relevant context, and specific examples for optimal results.

Iterative Refinement

Test different prompt variations and analyse model outputs, refining for clarity and precise intent.

Mastering Prompt Engineering

Common Mistakes to Avoid

Avoid vague prompts, excessive information, and lack of examples, as these reduce effectiveness and clarity.

Skill Development Resources

Explore guides and forums to expand prompt engineering expertise and stay updated with best practices.

Crafting Effective Prompts

Organise with Headings and Bullets

Use clear headings and bullet points to structure your prompts, making them easier to follow and understand.

Crafting Effective Prompts

Provide Clear Examples

Examples help clarify your instructions, ensuring your expectations are understood and followed accurately.

Crafting Effective Prompts

Avoid Common Mistakes

Steer clear of vague requests, inconsistent formatting, and unclear expectations to improve prompt effectiveness.

Crafting Effective Prompts

Encourage Actionable Responses

Structured, clear prompts foster precise, actionable answers from AI, benefiting both users and results.

Evolution of LLMs Across Domains

Domain	GPT-3 (2020)	GPT-4 (2023)	GPT-5 (2025)	Multi-Modal Models (2024–2025)
HR	Needed detailed prompts to generate job descriptions or policy drafts.	Role-based prompting produced refined job descriptions and interview questions.	Generates entire recruitment workflows, evaluates resumes, and suggests interview panels.	Analyze CVs (PDF) + Job Descriptions (text) and auto-generate candidate match reports.

Evolution of LLMs Across Domains

Domain	GPT-3 (2020)	GPT-4 (2023)	GPT-5 (2025)	Multi-Modal Models (2024–2025)
QA	Could create test cases with very explicit instructions but often incomplete.	Generates scenario-based test cases with better coverage.	Creates full regression test suites, predicts high-risk areas, and integrates defect analysis.	Reads logs, screenshots, or video test runs and generates detailed bug reports.

Evolution of LLMs Across Domains

Domain	GPT-3 (2020)	GPT-4 (2023)	GPT-5 (2025)	Multi-Modal Models (2024–2025)
Development	Produced small code snippets; required significant correction.	Creates structured, context-aware code with API integration and debugging help.	Generates production-grade code, test cases, and supports repository-wide refactoring.	From diagrams (image) + requirements (text), generates backend & frontend code scaffolding.

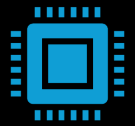
Evolution of LLMs Across Domains

Domain	GPT-3 (2020)	GPT-4 (2023)	GPT-5 (2025)	Multi-Modal Models (2024–2025)
Business Analysis (BA)	Summarized requirements but lacked structure.	Generates structured user stories, acceptance criteria, and process flows.	Handles end-to-end BA cycle: requirements → gap analysis → recommendations.	Converts meeting transcripts (audio) + workflow diagrams (images) into formal BRDs.

Applications Across Domains

1. Human Resources (HR)

2. Quality Assurance (QA)



3. Software Development (Dev)

4. Business Analysis (BA)

1. Human Resources (HR)

Prompt Engineering enables HR professionals

to automate and optimize daily tasks,

making people management more efficient.

HR Applications

Job Descriptions

Resume Screening

Interview Questions

Policy Drafting

Employee Engagement

Job Descriptions

GENERATE

ROLE-SPECIFIC JD

IN SECONDS.

Resume Screening

COMPARE

CANDIDATE RESUMES

WITH JD AND

HIGHLIGHT BEST MATCHES.

Interview Questions

CREATE COMPETENCY-BASED AND

BEHAVIORAL INTERVIEW QUESTIONS.

Policy Drafting

DRAFT HR POLICIES

LEAVE POLICY,

REMOTE WORK GUIDELINES,

PERFORMANCE EVALUATION

Employee Engagement

GENERATE NEWSLETTERS,

EVENT COMMUNICATIONS, OR

FEEDBACK FORMS.

Example Prompt:

“Act as an HR Manager.
Create an interview question set for a Data Analyst role focusing on problem-solving and communication skills.”

2. Quality Assurance (QA)

QA TEAMS BENEFIT FROM

AI-DRIVEN TEST GENERATION,

BUG DETECTION, AND

FASTER VALIDATION CYCLES.

2. QA Applications

Test Case Generation:

Create functional,

boundary,

negative test cases

automatically.

2. QA Applications

Bug Reporting:

Convert logs or error messages

into structured bug reports.

2. QA Applications

TEST AUTOMATION SUPPORT:

GENERATE SELENIUM,

PLAYWRIGHT,

PYTEST SCRIPTS.

2. QA Applications

Risk Coverage

Suggest

high-risk test areas

based on requirements.

2. QA Applications

REGRESSION TESTING:

CREATE

COMPREHENSIVE TEST SUITES

FROM USER STORIES.

Example Prompt:

“You are a QA engineer.
Generate 8 positive and 5 negative test cases
for an online shopping cart checkout process.”

3. Software Development (Dev)

DEVELOPERS CAN USE PROMPT ENGINEERING

TO ACCELERATE CODING, DEBUGGING,

AND DOCUMENTATION.

Software Development Applications

Code Generation:

Produce code snippets,

APIs, and full modules.

Software Development Applications

Debugging:

Explain

Error messages

Suggest fixes

Software Development Applications

Documentation:

Generate

API documentation

README files

Usage examples

Software Development Applications

Code Review:

Suggest improvements in

Performance,

Security,

Readability.

Software Development Applications

Architecture Guidance:

Draft

System Design Patterns

Refactoring strategies.

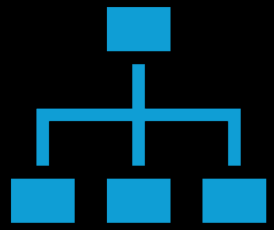
Example Prompt

“Write a FastAPI endpoint
to update customer data in a PostgreSQL database,
including input validation and error handling.”

4. Business Analysis (BA)

Business Analysts can transform

raw requirements into



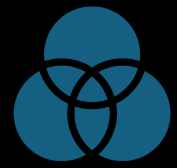
structured deliverables faster.

4. Business Analysis Applications

User Stories & Epics:

Generate

JIRA-style user stories



with acceptance criteria.

4. Business Analysis Applications

Requirement Summaries:

Convert

Meeting notes into

Structured Requirement documents.

4. Business Analysis Applications

Process Flow Documentation:

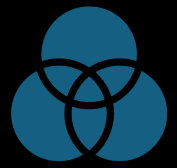
Suggest

workflow diagrams

SOPs.

4. Business Analysis Applications

Gap Analysis:



Compare

current vs. desired

system capabilities.

4. Business Analysis Applications

Business Case Creation:

Draft

ROI analysis

Feature prioritization documents.

Prompt Example

“You are a Business Analyst.

Convert the following notes into three user stories with acceptance criteria:

customers should log in with email/mobile,

view order history, and download invoices.”

Key Takeaway

Prompt Engineering isn’t limited to developers

it empowers HR to manage talent,

QA to ensure quality,

Developers to code efficiently, and

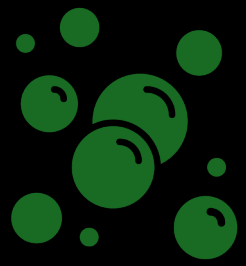
BAs to structure business requirements.

Key Takeaway

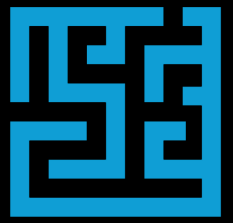
EACH DOMAIN CAN **SAVE TIME, REDUCE ERRORS, AND SCALE PRODUCTIVITY** WITH EFFECTIVE PROMPT DESIGN.

Prompt types

Zero-shot,



Few-shot,



Chain-of-Thought

Zero-Shot Prompting

Directly asking the model

to perform a task

without providing any examples.

Zero-Shot Prompting

Strength:

Fast, simple,

useful when tasks are straightforward.

Zero-Shot Prompting

Limitation:

May lack accuracy

if the task is complex.

Examples Across Domains

HR: *“Write a job description for a Machine Learning Engineer.”*

QA: *“Generate 5 test cases for login functionality.”*

Development: *“Write a Python function to reverse a string.”*

BA: *“Summarize requirements for an online food ordering app in bullet points.”*

2. Few-Shot Prompting

Providing

few examples in the prompt

to guide the model’s response

style and format.

2. Few-Shot Prompting

Strength:

Increases accuracy and consistency.

2. Few-Shot Prompting

Examples Across Domains

Surendra Panpaliya
Founder and CEO, GKTCS Innovations
https://www.linkedin.com/in/surendrarp

HR

“Here are examples of interview questions:

Tell me about a time you resolved a workplace conflict.

How do you prioritize tasks under pressure?

Now generate 3 more behavioral interview questions.”

QA

“Example test case:

Verify that valid login credentials allow access to the dashboard.

Example test case:

Verify that invalid credentials display an error message.

Generate 3 more test cases for login functionality.”

Development

Example code snippet:

```
# Add two numbers
def add(a, b):
return a + b
```

Generate a function to multiply two numbers in the same style.

BA

Example User Story:

As a customer, I want to log in with my email so that I can access my profile.

As a customer, I want to reset my password so that I can regain access if I forget it.

Create 2 more user stories for online shopping.

Chain-of-Thought (CoT) Prompting

Asking the model to

“show its reasoning”

step by step

before giving the final answer.

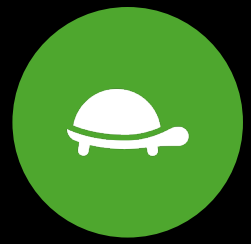
Chain-of-Thought (CoT) Prompting

STRENGTH:

IMPROVES REASONING FOR COMPLEX TASKS.

LIMITATION:

LONGER RESPONSES,



MAY BE SLOWER.

HR

“Explain step by step
how you would evaluate a resume
for a Data Scientist role,
then provide a final decision
(shortlist or reject).”

QA

Think step by step.

First list possible input values for a shopping cart checkout.

Then explain expected outputs.

Finally, write test cases.

Development

Think step by step.
First explain the logic
to check if a number is prime.
Then write Python code that implements it.

BA

Step by step:

1) Identify stakeholders,

2) Capture requirements,

3) Prioritize features.

Now convert these into 3 user stories with acceptance criteria.

Key Takeaway

Zero-Shot:

Quick tasks, no examples needed.

Few-Shot:

Guided tasks, better accuracy with examples.

Key Takeaway

Chain-of-Thought:

Complex reasoning tasks,

requires step-by-step explanation.

Understanding Prompt Structure

Key components of a prompt

(Instruction, Context, Example, Constraint).

Format styles: questions, templates, structured tasks.

Prompting for **reasoning vs. factual accuracy**.

Key Components of a Prompt

Instruction

Context

Example(s)

Constraint(s)

Instruction

The direct task
you want the AI to perform.
Clear, action-oriented wording.

Context

Background information
to help the AI understand the situation.
Provides relevance and domain alignment.

Example(s)

Demonstrations of the desired style,
format, or structure.
Helps guide the AI’s output.

Constraint(s)

Rules or limitations for the output.

Examples: word limits, tone, format, or specific inclusions/exclusions.

Human Resources (HR)

- **Instruction:** *“Write an interview question set.”*
- **Context:** *“The role is for a Data Analyst in the BFSI domain.”*
- **Example:** *“Example question: How would you validate financial datasets for accuracy?”*
- **Constraint:** *“Provide 5 questions, each focusing on problem-solving skills.”*

HR Prompt Example

Write 5 interview questions for a
Data Analyst in the BFSI domain.
Example:
How would you validate financial datasets for accuracy?
Keep the questions problem-solving oriented.

Quality Assurance (QA)

- **Instruction:** *“Generate test cases.”*
- **Context:** *“The application is an e-commerce checkout system.”*
- **Example:** *“Example test case: Verify that valid credit card details allow successful payment.”*
- **Constraint:** *“Provide at least 3 positive and 3 negative test cases.”*

Full Prompt

Generate test cases for

an e-commerce checkout system.

Example:

Verify that valid credit card details allow successful payment.
Provide at least 3 positive and 3 negative cases.

Development (Dev)

- **Instruction:** *“Write a Python function.”*
- **Context:** *“The function should interact with a PostgreSQL database for customer data.”*
- **Example:** *“Example: A function to fetch customer by ID with error handling.”*
- **Constraint:** *“Use FastAPI framework and include docstrings.”*

Full Prompt

Write a FastAPI endpoint in Python
that updates customer data in a PostgreSQL database.
Example:
fetching customer by ID with error handling.
Use FastAPI and include docstrings.

Business Analysis (BA)

- **Instruction:** *“Create user stories.”*
- **Context:** *“The project is an online food delivery application.”*
- **Example:** *“Example: As a customer, I want to log in with my email so that I can view my past orders.”*
- **Constraint:** *“Generate 3 user stories with acceptance criteria in Gherkin format.”*

Full Prompt

Create 3 user stories for an online food delivery app.

Example:

As a customer, I want to log in with my email so that I can view my past orders.

Add acceptance criteria in Gherkin format.

Key Takeaway

- Every effective prompt is made of:
- **Instruction → What to do**
- **Context → Where/why it applies**
- **Example → How the output should look**
- **Constraint → Rules to follow**

Exploring Prompt Format Styles

Question-Based Prompts

Template-Based Prompts

Structured Task Prompts

Question-Based Prompts

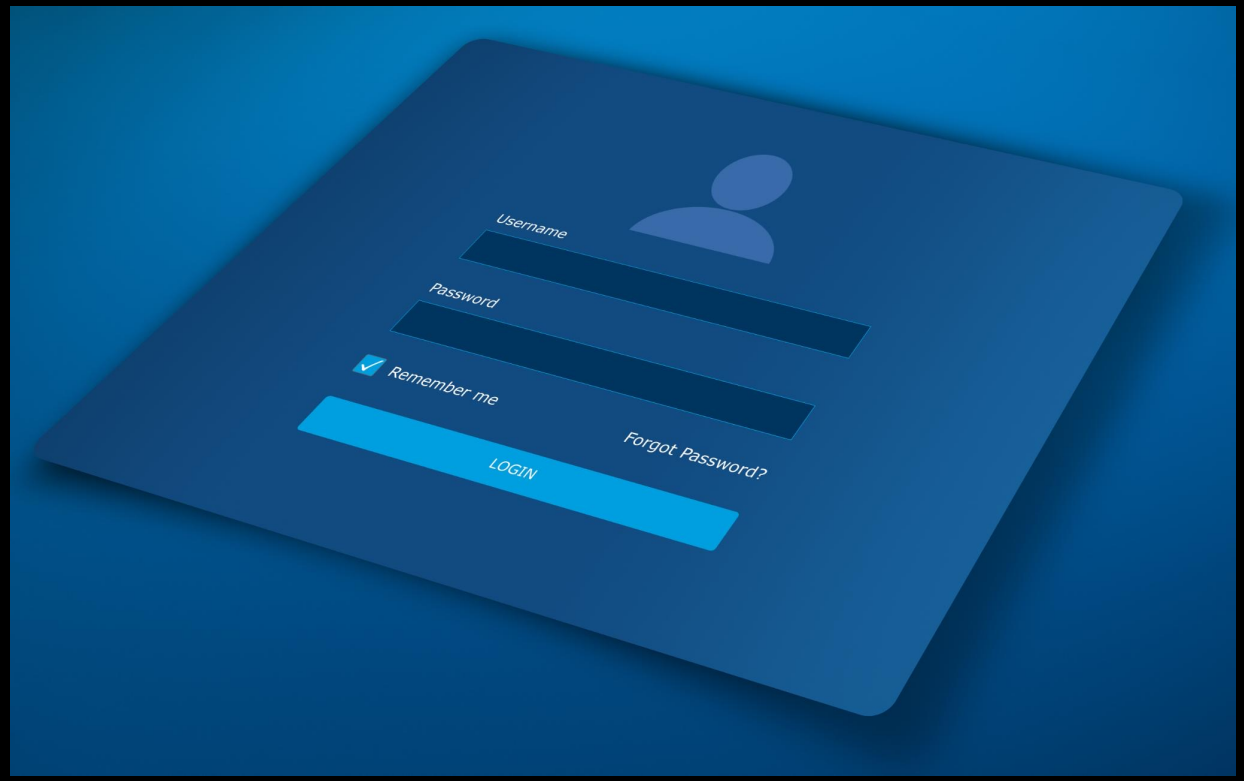
- Frame prompts as direct questions to get precise answers.
- Simple and intuitive style ideal for quick information gathering.
- Effective for exploring ideas and clarifying specific details.
- Encourages concise and focused responses from AI models.

Examples

- **HR:** *“What are 5 behavioral interview questions for a Project Manager role?”*
- **QA:** *“What test scenarios should be considered for a password reset feature?”*
- **Development:** *“How do I write a Python function to validate an email address?”*
- **BA:** *“What are the key requirements for an online ticket booking system?”*

Template-Based Prompts

- Use predefined structures or placeholders for consistency.
- Ideal for repetitive tasks requiring uniform output formats.
- Helps maintain quality and standardisation across responses.
- Facilitates easier automation and scaling of prompt use.



Examples

HR:

“Create a job description for [Job Title] in the [Industry] domain. Include: Responsibilities, Skills, Qualifications.”

QA:

“Write test cases for [Feature]. Include: Test ID, Test Step, Expected Result.”

Examples

- **Development:**
- *“Generate a [Language] function to [Task]. Ensure: Input validation, Error handling, Example usage.”*
- **BA:**
- *“Convert the following requirement into a user story: [Requirement]. Use format: As a [user], I want [goal], so that [reason].”*

Structured Task Prompts

- Break prompts into clear steps or sections.
- Combine instructions, context, and constraints for clarity.
- Best suited for complex workflows needing detailed outputs.
- Enhances AI's ability to handle multi-part and nuanced tasks.

Examples:

- **HR:**
- *“Step 1: List 3 key skills needed for a Data Analyst.*
- *Step 2: Draft 2 interview questions for each skill.*
- *Step 3: Format as a table.”*

QA

- *“Step 1: Identify input fields for the registration form.*
- *Step 2: Create positive and negative test cases.*
- *Step 3: Provide expected results.”*

Development

“Step 1: Explain the logic for calculating factorial.

Step 2: Write Python code for factorial.

Step 3: Provide test cases with expected outputs.”

BA

“Step 1: Identify stakeholders for a loan approval system.

Step 2: Write 3 user stories.

Step 3: Add acceptance criteria for each user story.”

Key Takeaway

Questions → Quick answers, idea exploration.

Templates → Consistent, repeatable outputs.

Structured Tasks → Complex, multi-step solutions with clarity.

Reasoning vs Factual Accuracy

Value of In-Depth Reasoning

Encouraging reasoning develops critical thinking and deeper understanding, allowing for nuanced insight and thoughtful analysis.

Reasoning vs Factual Accuracy

Importance of Factual Accuracy

Factual accuracy focuses on providing correct, verified details, reducing errors and maintaining informational reliability.

Reasoning vs Factual Accuracy

Balancing Both Approaches

Combining robust reasoning with factual accuracy leads to better understanding and trustworthy results, crucial in academic and professional settings.

Reasoning prompts

Goal: correct thinking and decisions.
Style: ask for steps, assumptions,
trade-offs, edge cases, and
a justified conclusion.

Reasoning prompts

- Guardrails: require structure
- (Steps → Checks → Decision),
- Ask for alternatives and failure modes.

Factual-accuracy prompts

- Goal: correct facts and references.
- Style: provide (or request) sources,
- versions, dates; forbid invention;
- ask for citations and uncertainty flags.

Factual-accuracy prompts

Guardrails:

“Use only the provided context,”

“Cite sources with name + date,”

“If unknown, say ‘Not found’.”

2) Generic templates you can reuse

Reasoning-first (generic)

You are a senior {role}.

Task: {decision/problem}.

Constraints: {business rules/limits}.

2) Generic templates you can reuse

Produce:
1) Assumptions (explicit),
2) Step-by-step reasoning,
3) Options with pros/cons,
4) Edge cases & risks,
5) Final recommendation with rationale.
Keep it under {length}.

Fact-first (generic)

You are a meticulous researcher.

Task: {fact lookup/summary}.

Sources: {paste links or attach text}.

Use only these sources.

Fact-first (generic)

Output:

- Bullet facts with source name + date,
- Exact figures with units,
- A short “Unknown/Needs SME” list for gaps.

No speculation.

If a fact is missing, write “Not in sources.”

3) Domain examples

(Reasoning vs. Factual)

A) Human Resources (HR)

Reasoning-first (candidate fit)

You are an HR Business Partner.

Task: Assess candidate fit for “Data Analyst – BFSI”.

Inputs: JD (skills: SQL, Python, risk analytics), Resume (attached).

Constraints: Prioritize regulated-domain experience and stakeholder communication.

A) Human Resources (HR)

Deliver:

1) Assumptions,
2) Scored fit across must-have vs nice-to-have (weightings visible),
3) Risk flags (skill or domain gaps),
4) Recommend Hire/No-Hire with rationale.

Fact-first (policy facts)

- You are an HR policy researcher.
- Task: Summarize statutory leave entitlements relevant to our India office.
- Sources: [attached policy PDFs, government page printouts].

Fact-first (policy facts)

- Deliver:
- - Table: Leave type | Min days | Source name | Publication date.
- - Cite each row. If a value isn’t in the sources, write “Not in sources”.
- No interpretation beyond cited text.

B) Quality Assurance (QA)

Reasoning-first (test strategy)

You are a QA Lead.

Task: Create a test strategy for checkout (cards, UPI, coupons, PII).

Constraints: Go-live in 2 weeks; critical risks are payment failures and double charges.

B) Quality Assurance (QA)

Deliver:

1) Risk-based prioritization (High/Med/Low) with reasons,
2) Test design: functional, boundary, negative, integration,
3) Data strategy and observability checks,
4) Exit criteria with measurable thresholds,
5) Top 10 edge cases and how to catch them.

Fact-first (standards & references)

You are a QA researcher.

Task: List accessibility test requirements applicable to web checkout.

Sources: [WCAG 2.2 quick ref PDF], [company accessibility standard].

Fact-first (standards & references)

Deliver a checklist:

- Requirement | Brief test | Source (name + section) | “Pass/Fail/NA” placeholder.

No items beyond cited sources.

C) Development (Software Engineering)

- **Reasoning-first (architecture choice)**
- You are a Principal Engineer.
- Task: Choose between monolith, services, or event-driven for an order system at 10k TPS.
- Constraints: Team of 6; 99.9% availability; data consistency > latency.

C) Development (Software Engineering)

Deliver:

1) Assumptions & workload model,
2) Option analysis (throughput, failure modes, complexity),
3) Data consistency strategy,
4) Observability/operability plan,
5) Final choice with trade-offs and “if we’re wrong” rollback plan.

Fact-first (API correctness)

You are a documentation-strict developer.

Task: Write a FastAPI endpoint for upserting customer records.

Sources: [FastAPI docs excerpt], [internal API spec v1.3], [PostgreSQL docs excerpt].

Fact-first (API correctness)

Deliver:

- Code complying with the spec,
- Inline comments citing the section (e.g., “Spec v1.3 §4.2”),
- A short note listing any spec ambiguities as “Unknown”.

Do not use any behavior not present in the sources.

D) Business Analysis (BA)

Reasoning-first (prioritization)

You are a Senior BA.

Task: Prioritize backlog items for MVP of a loan origination portal.

Inputs: 12 backlog items with brief business value notes.

Constraints: MVP in 8 weeks; compliance items are non-negotiable.

D) Business Analysis (BA)

Deliver:

1) Assumptions,
2) RICE scoring (show formula and inputs),
3) Dependencies & risks,
4) Final ranked list with cutline for MVP scope.

Fact-first (market & KPI facts)

- You are a BA researcher.
- Task: Compile current KPIs definitions used by our company for “Lead-to-Loan”.
- Sources: [internal KPI handbook PDF], [analytics glossary], [last quarter’s KPI deck].
- Deliver:
- - KPI | Precise formula | Data source of truth | Reporting cadence | Source citation.
- No estimates; if unspecified, mark “Not in sources”.

4) How to combine both (two-pass workflow)

- **Reasoning pass**: plan, decide, or design (ask for assumptions, steps, edge cases).
- **Fact check pass**: verify all claims against provided sources (ask for citations, dates, and “Unknown” flags).
- **Final pass**: concise decision or artifact, noting any residual uncertainties.

Prompt scaffold:

- Pass 1 (Reasoning): Produce plan with assumptions, options, risks.
- Pass 2 (Fact Check): Validate each assumption against [sources]. Add citations or mark “Not in sources”.
- Pass 3 (Final): Provide the final recommendation/artifact;
- list residual unknowns and next data requests.

5) Quick checklists

- **Reasoning prompts**
- Ask for: assumptions, steps, alternatives, edge cases, decision criteria.
- Require: explicit trade-offs, measurable exit criteria, and a short final recommendation.

5) Quick checklists

- **Factual prompts**
- Provide/require sources, versions, dates.
- Forbid speculation; allow “Unknown/Not in sources.”
- Ask for citations per fact and mark time-sensitive data.

Hands-On Step by Step: Crafting Effective Prompts

- Activity: Basic vs. improved prompts.
- Role prompting, style prompting, task chaining.
- Tools: **ChatGPT, OpenAI Playground**.
- https://platform.openai.com/chat/edit?prompt=pmpt_68af3c8d896481908f1db839ddc561c90db33e8a1d08c679&version=1

ChatGPT

Use a fresh chat for each activity.

Start with a short “system instruction” like:

“You are a precise, enterprise-grade assistant.

Prefer structured, verifiable outputs.”

OpenAI Playground

Pick a chat-capable model.
Set a **System** message for role,
adjust **Temperature**
(0.2–0.5 for accuracy; 0.7–1.0 for creativity), and
keep Max Output high enough for your task.
The Playground lets you experiment with roles
and parameters interactively.

Case Study (HR 1)

Automating Job Description Drafting

1. Background

- The HR team at a **mid-size IT services firm** spends significant time manually drafting job descriptions (JDs).
- Each JD takes ~2 hours to create.
- Inconsistencies occur across roles because different HR managers use varied styles.
- Business leaders often ask for faster turnaround for urgent hiring needs.

1. Background

- **Goal:** Use prompt engineering with LLMs (e.g., ChatGPT, OpenAI Playground) to automate and standardize JD creation.

2. Challenges

- JDs lacked standard structure (some had responsibilities only, others missed qualifications).
- Language was sometimes too generic and not tailored to domain (e.g., “Developer” vs. “Developer in BFSI risk analytics”).
- Scaling across multiple job roles took excessive time.

3. Solution: Role-Based Prompt Engineering

- **Step 1: Basic Prompt (Inefficient)**
- *“Write a job description for a Data Analyst.”*

Step 2: Improved Role-Based Prompt

- You are an HR Business Partner at a BFSI firm.
- Task: Draft a job description for “Data Analyst – Risk Analytics”.
- Context: Candidate should have experience with SQL, Python, and regulatory reporting.
- Constraints: Include clear sections: Job Summary, Key Responsibilities, Required Skills, Preferred Skills, Qualifications.
- Style: Professional and engaging, suitable for posting on LinkedIn.

Output

Clear job summary aligned to BFSI domain.

Well-structured bullets for responsibilities and skills.

Ready-to-publish LinkedIn JD with minimal editing.

4. Results

- **Time Saved:** JD drafting reduced from ~2 hours to ~15 minutes (including edits).
- **Consistency:** All JDs now follow a uniform structure.
- **Scalability:** HR team can generate 10–15 JDs/day across roles.
- **Quality:** Hiring managers rated the AI-assisted JDs as “90% ready-to-use.”

5. Key Learning for HR Teams

- Prompt structure matters: **Instruction + Context + Constraints** = high-quality JD.
- Role-based prompts make AI “think like an HR professional,” not a generic assistant.
- Human oversight is still needed for fine-tuning language and compliance, but efficiency gains are significant.

Case Study (HR 2)

Resume screening & shortlisting with ethical safeguards.

1. Background

- A **global IT consulting firm** received **1,200+ resumes** for a single *Data Analyst – BFSI* position.
- Manual screening took HR teams 2–3 weeks.
- Inconsistent evaluation criteria led to complaints from hiring managers.
- Leadership wanted a **faster, fairer, and transparent** resume shortlisting process.

1. Background

- **Objective:** Use **Prompt Engineering + LLMs** to automate first-level screening while ensuring **ethical safeguards**.

2. Challenges

- **Bias Risks** – Resumes included gender, age, photos, and personal details that could lead to unconscious bias.
- **Inconsistency** – Different recruiters applied varied standards.
- **Scalability** – Hundreds of resumes per role, across multiple geographies.
- **Transparency** – Hiring managers wanted clear reasoning behind shortlisting/rejections.

3. Solution: Role-Based Ethical Prompting

- **Prompt Design**
- You are an unbiased HR Recruiter.
- Task: Screen and evaluate the candidate resume against the attached job description.

3. Solution: Role-Based Ethical Prompting

Constraints:

1. Focus only on skills, experience, and qualifications relevant to the JD.

2. Ignore personal identifiers such as gender, age, marital status, or photo.

3. Solution: Role-Based Ethical Prompting

3. Output must include:
   - Fit Score (0–100)
   - Strengths (skills & experiences matching the JD)
   - Gaps (missing or weak areas)
   - Final Decision: Shortlist / Reject with rationale.
4. Style: Neutral, professional, compliance-friendly.

4. Ethical Safeguards

- **Bias Mitigation:** Explicitly instructed the AI to ignore gender, age, and personal details.
- **Transparency:** Every decision included Fit Score + Strengths + Gaps + Rationale.
- **Consistency:** Uniform template applied across 500+ resumes.
- **Human Oversight:** Recruiters validated AI outputs before finalizing shortlists.

5. Results

- **Time Savings:** Screening time reduced from 20 minutes/resume → 3 minutes/resume.
- **Scalability:** 1,200 resumes screened in <1 day.
- **Fairness:** All candidates evaluated only on skills/experience.
- **Adoption:** Hiring managers trusted the structured scoring, improving collaboration with HR.

6. Key Learnings

- Prompt Engineering can **standardize** resume screening across recruiters.
- Embedding **ethical safeguards in prompts** reduces discrimination risk.
- Transparency (Fit Score + Rationale) builds trust with business stakeholders.
- Final decision must remain with humans → AI acts as **decision-support, not decision-maker**.

Let’s Connect

Professional Email Contact

surendra@gktcs.com

LinkedIn Networking

https://www.linkedin.com/in/surendrarp

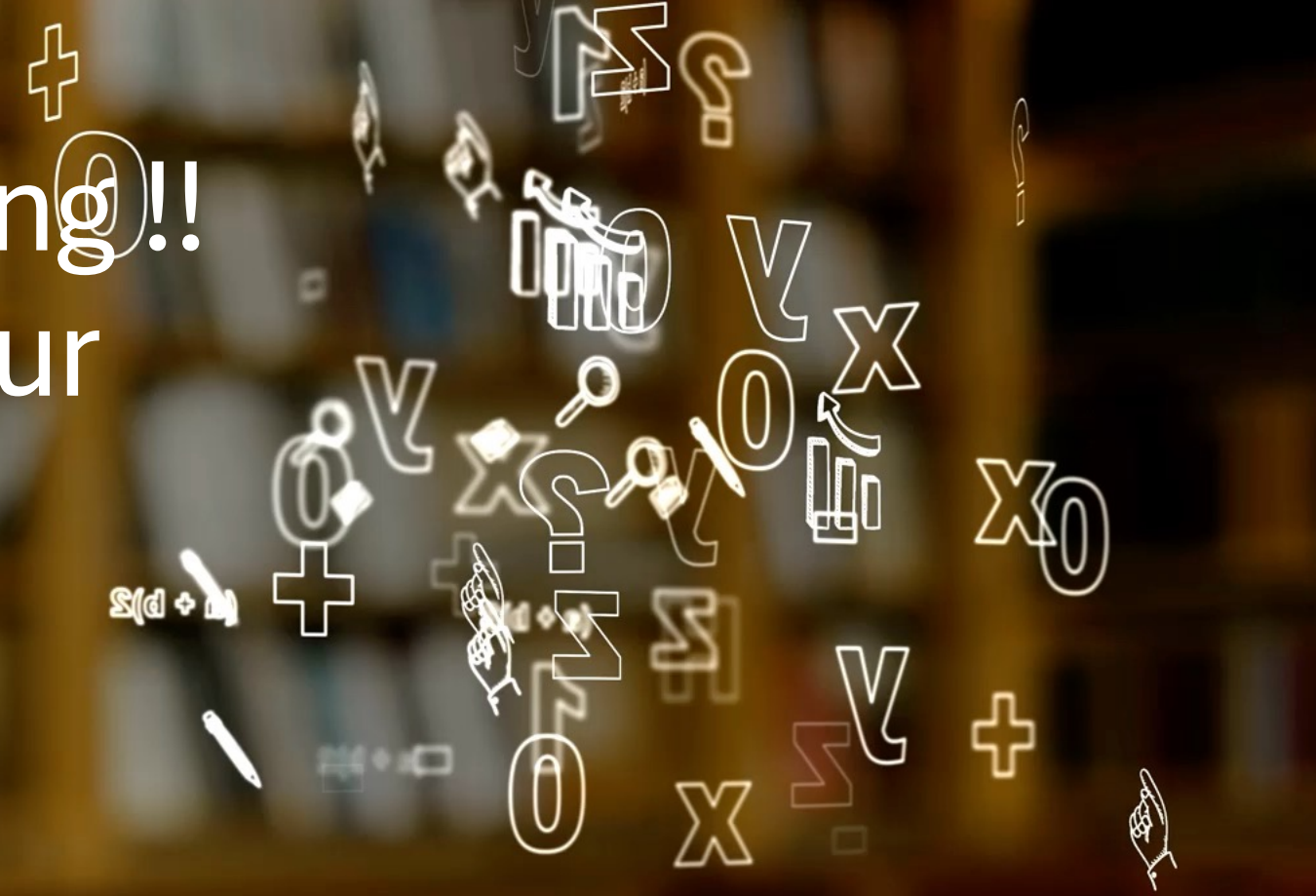
Company Website Information

https://www.gktcs.com

Direct Phone Assistance

+91 9975072320

Happy Learning!! Thanks for Your Patience ☺



Surendra Panpaliya

GKTCS Innovations