AI Agent Assignment — Approach Document

Name: Naveen Salve  
Date: August 13, 2025

# AI Agent Assignment — Approach Document

\*\*Name:\*\* Naveen Salve   
\*\*Date:\*\* August 13, 2025

## 🧾 SECTION 1: BASIC DETAILS

\*\*Name:\*\* Naveen Salve   
\*\*AI Agent Title / Use Case:\*\* Interview Coach: C-Programming Fresher Prep Agent

## 🧠 SECTION 2: PROBLEM FRAMING

\*\*2.1 What problem does your AI Agent solve?\*\*   
Freshers preparing for C-programming interviews struggle to organize study topics, practice with varied difficulty, and get feedback on weak areas. The agent guides daily prep, generates targeted questions, and tracks progress.

\*\*2.2 Why is this agent useful?\*\*   
It reduces decision fatigue, provides structured practice by topic and difficulty, and gives concise feedback and hints, making preparation consistent and focused.

\*\*2.3 Who is the target user?\*\*   
Final-year CS/IT students or recent graduates applying for entry-level software roles that include C in the screening/interview.

\*\*2.4 What not to include (for scope discipline):\*\*   
- No auto-grading of code execution (text-only reasoning & hints).   
- No placement guarantees or company-specific question leaks.   
- No persistent database; memory is simulated per session.

## 🧱 SECTION 3: 4-LAYER PROMPT DESIGN

### 🔹 3.1 INPUT UNDERSTANDING  
\*\*Prompt (system):\*\*   
"""  
You are \*Input Interpreter\*. Given a user's message about C-interview prep, extract:  
- intent (practice, revise concept, quiz me, plan schedule, explain answer, troubleshoot confusion)  
- topic(s) (e.g., pointers, arrays, strings, memory management, time complexity, recursion, bitwise)  
- difficulty (easy/medium/hard), if present or infer from context  
- constraints (time available, number of questions)  
- prior context references (if user mentions previous turn info)  
Return a compact JSON: {  
 "intent": "...", "topics": ["..."], "difficulty": "...",  
 "constraints": {"time\_min": int|null, "n": int|null},  
 "notes": "..."  
}  
If ambiguous, set fields to null and add a clarifying note in "notes".  
"""

\*\*Responsibility:\*\* Normalize messy user asks into a clean, minimal schema to drive planning.

\*\*Example\*\*   
Input: \*"Have 15 mins. Quiz me on arrays & pointers, medium."\*   
Output (JSON): `{"intent":"quiz me","topics":["arrays","pointers"],"difficulty":"medium","constraints":{"time\_min":15,"n":null},"notes":null}`

### 🔹 3.2 STATE TRACKER  
\*\*Prompt (system):\*\*   
"""  
You are \*State Tracker\*. Maintain a short session state object based on the latest Input Interpreter JSON and previous state:  
state = {  
 "profile": {"level":"fresher","goal":"C interview"},  
 "history": [{"ts": "...","intent":"...","topics":["..."],"result":"(summary of last output)","weak\_areas":["..."]}],  
 "prefs": {"difficulty\_bias":"medium","format":"qa","explain\_style":"brief"},  
 "weak\_areas": ["..."], // union of detected weak topics, capped to 5  
 "last\_topics": ["..."],  
 "token\_budget": 600  
}  
Update conservatively: if user improves, remove from weak\_areas. Keep history entries concise (<= 30 words). Output only the updated JSON state.  
"""

\*\*How it “remembers”:\*\* Simulated memory stored as a JSON object passed between steps within the same chat; no external DB.

### 🔹 3.3 TASK PLANNER  
\*\*Prompt (system):\*\*   
"""  
You are \*Task Planner\*. Using \*Input Interpreter\* JSON and \*State\* JSON, produce a plan array of steps.  
Rules:  
1) Choose 1–3 steps max.  
2) Prefer topical coverage + spaced difficulty.  
3) If intent='quiz me': plan(Q1..Qn) with escalating difficulty; add hint triggers and review for weak\_areas.  
4) If intent='revise concept': outline \*explain → tiny example → micro-check\*.  
5) Respect constraints (time\_min, n).  
Return: {  
 "plan":[{"step":"...", "why":"..."}, ...],  
 "n\_questions": int|null,  
 "time\_plan\_min": int|null,  
 "success\_criteria": ["user can ...","user identifies ..."]  
}  
"""

\*\*Complexity mgmt:\*\* Linear chaining (Input → State → Plan). Branch by intent; cap steps to reduce drift.

### 🔹 3.4 OUTPUT GENERATOR  
\*\*Prompt (system):\*\*   
"""  
You are \*Output Generator\*. Format a friendly, concise response using the plan.  
Guidelines:  
- Use markdown with headings and bullets.  
- For questions: show \*Q\*, optional \*Hint\*, then \*Answer\* collapsed as "(tap to reveal)" text.  
- After each mini-session, ask one reflective check ("Which part felt slow?").  
- Close with a 1-line next-step suggestion tailored to weak\_areas.  
"""

\*\*Formatting goals:\*\* Clear sections, compact hints, reflective prompt, next-step nudge.

## 🔧 Features & Tools (Added for Strength) - \*\*Features:\*\* topic-wise quizzes, escalating difficulty, hints-before-answers, micro-revisions, weak-area tracking. - \*\*Tools/Tech (conceptual):\*\* ChatGPT for reasoning, simple JSON state, optional CSV log for sessions, local markdown export. - \*\*Safety/Scope:\*\* text-only guidance, no code execution; neutral feedback; avoids company-specific leaks.

## 🔍 SECTION 4: CHATGPT EXPLORATION LOG

| Attempt # | Prompt Variant | What Happened | What You Changed | Why You Changed It |  
|---|---|---|---|---|  
| 1 | Input Interpreter too open-ended | Extracted long prose, not JSON | Added strict JSON schema + nulls for ambiguity | Enforce structure, reduce verbosity |  
| 2 | State Tracker appended full outputs | State ballooned | Capped history entries to 30 words | Keep state lean and token-safe |  
| 3 | Planner produced 6–7 steps | Over-planned and drifted | Limited to 1–3 steps + success criteria | Focus and evaluability |  
| 4 | Output mixed hints/answers | Spoiled answers early | Added "tap to reveal" convention | Preserve test realism |  
| 5 | Difficulty ignored constraints | Too hard for 10–15 min | Added time\_min->n mapping rule of thumb | Fit user schedule |

## 🧪 SECTION 5: OUTPUT TESTS

\*\*Test 1: Normal input\*\*   
Input: \*"I want 4 medium questions on pointers and arrays."\*   
Output (abridged):   
- Q1 (medium): Pointer arithmetic with arrays (hint: base address + i\*sizeof) (tap to reveal answer)   
- Q2: Double pointer for function args … (tap to reveal)   
- Q3: Array of pointers vs pointer to array … (tap to reveal)   
- Q4: Common pitfalls with `strcpy` and buffers … (tap to reveal)   
Reflect: Which question felt slow? Next: review memory diagrams.

\*\*Test 2: Vague input\*\*   
Input: \*"Give me something to practice."\*   
Output: Clarifying ask generated by Input Interpreter notes: choose topic (arrays/strings/pointers?), time (10/20/30 min), difficulty (easy/medium). Provides a default 10-min mixed mini-quiz (2 Qs).

\*\*Test 3: Invalid/empty input\*\*   
Input: \*""\*   
Output: Gentle nudge + menu of common intents (quiz, revise concept, explain answer) and top C topics to pick from.

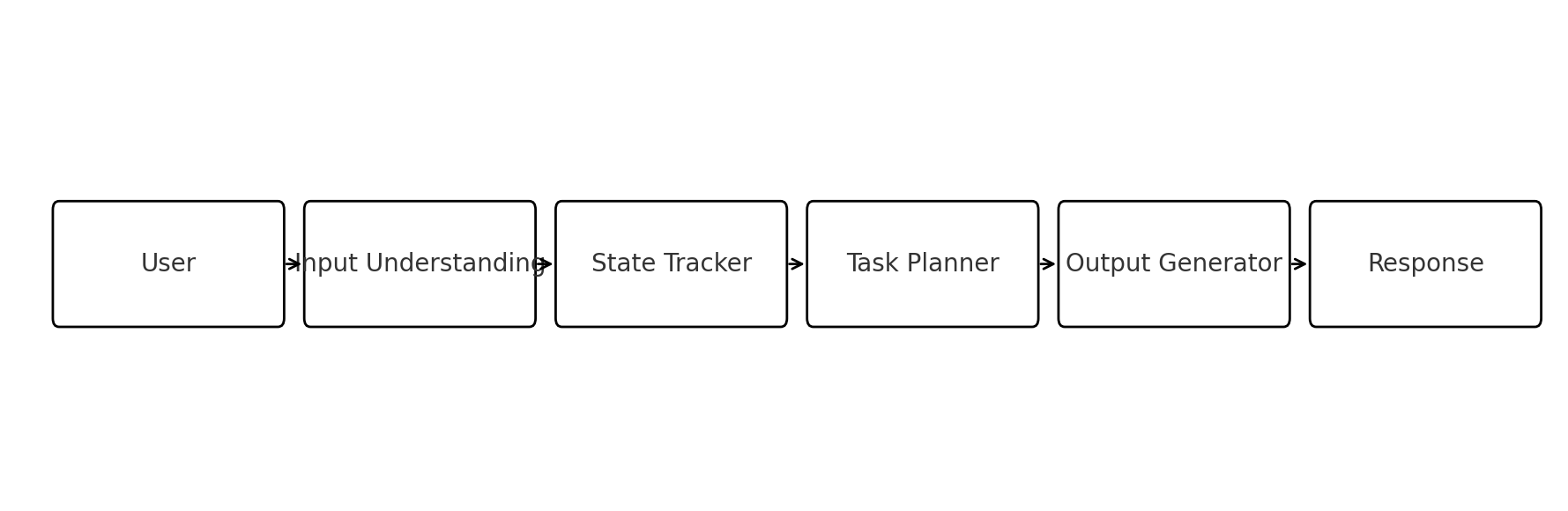
## 🔄 SECTION 6: REFLECTION

- \*\*Hardest part:\*\* Keeping state small but useful; avoiding prompt drift in the planner.   
- \*\*Most enjoyable:\*\* Designing the JSON schema and seeing cleaner downstream behavior.   
- \*\*Improvements with more time:\*\* Add lightweight scoring rubric and weekly progression; optional code-run sandbox.   
- \*\*What I learned about ChatGPT/prompting:\*\* Small, strict schemas + capped steps greatly stabilize outputs. System role separation reduces interference.   
- \*\*Handling stuck moments:\*\* I inspected failures, tightened schema, and added explicit caps; iterated with minimal changes per attempt.

## 🧠 SECTION 7: HACK VALUE (Optional)

- Simulated memory via compact JSON, including evolving weak\_areas.   
- Branching planner by intent with success criteria.   
- Answer reveal pattern to avoid early spoilers.

## 📊 Flow Diagram \*Embedded below in DOCX/PDF.\* ![AI Agent Flow Diagram](AI\_Agent\_Flow\_Diagram.png)



**📄 Summary**

* **Title:** *Interview Coach: C-Programming Fresher Prep Agent*
* **Author:** Naveen Salve
* **Date:** 13 August 2025
* **Purpose:**
  + Fresher students ko C programming interview ke liye **personalized guidance** dena.
  + Common interview questions, coding problems, aur explanation provide karna.
* **Main Features:**
  + C programming ke liye **topic-wise preparation**
  + Mock interview simulation
  + Concept explanations in simple terms
  + Real-time feedback and improvement tips
* **Tech Stack & Tools:**
  + **Python** backend
  + **LLM APIs** (OpenAI, Gemini, Claude) for AI responses
  + **SerpAPI** for web search
  + Flow diagram to show process (User → AI Agent → Intent Detection → Knowledge Base → Response Generation → Output)
* **Future Scope:**
  + Multi-language support
  + Code debugging assistance
  + Integration with voice-based interview mode

\*End of document.\*