StudyBuddy — AI Agent Development Challenge

Course Name: Growth OS AI Agent Development Challenge Student: Aaryaveer Yadav | Registration No: 23BAI10323 AI Agent Assignment Approach Document StudyBuddy Exam Revision Assistant

SECTION 1: BASIC DETAILS Name: Aaryaveer Yadav AI Agent Title / Use Case: StudyBuddy AI

Agent to help college students revise for short-answer exams

SECTION 2: PROBLEM FRAMING 2.1 What problem does your AI Agent solve? Many college

students struggle to convert long lecture notes into short, testable study items. StudyBuddy creates summaries, flashcards, and short quizzes from a topic or pasted notes. 2.2 Why is this agent useful? Reduces friction between raw notes and active-recall materials. Saves time and helps focused practice. 2.3 Target user: College students preparing for short-answer/recall-style exams. 2.4 What not to include: No long-form tutoring, no calendar integration, no full lecture generation.

SECTION 3: 4-LAYER PROMPT DESIGN

3.1 INPUT UNDERSTANDING Prompt (summary): You are the Input Understanding module for StudyBuddy. Extract topic, intent (want_quiz/want_summary/want_flashcards), difficulty, length, notes, and ask a clarifying question if needed. Output JSON: {"topic": "...", "intent": "...", "difficulty": "...", "length": n, "notes": "...", "clarifying_question": null} 3.2 STATE TRACKER Prompt (summary): Maintain a short context object with session_id, last_topic, last_intent, known_user_prefs, recent_items. Update it given Input JSON and return state JSON. Rules: inherit difficulty if omitted, FIFO recent_items size 5. 3.3 TASK PLANNER Prompt (summary): Given Input JSON and state, produce a step-by-step plan (JSON list) with actions like generate_summary, generate_quiz, generate_flashcards, or ask_for_notes. Include substeps for validation. 3.4 OUTPUT GENERATOR Prompt (summary): Produce the final markdown response. Rules for formatting: 2-sentence summary if requested, numbered flashcards (Q/A), numbered quiz with Answers section, Next steps, follow-up options ("make harder" / "more like this"). Tone: encouraging, concise. Limit 300400 words.

SECTION 4: CHATGPT EXPLORATION LOG Attempt 1: One-shot combined prompt -> unstructured

output. Change: modularize into 4 layers. Attempt 2: Added JSON Input Understanding -> better extraction but planner ambiguous. Change: explicit Task Planner JSON. Attempt 3: Full 4-layer flow

-> best results; tuned Output Generator for word limits and follow-up options.

SECTION 5: OUTPUT TESTS Test 1 - Normal: Input: "I want a 5-question medium quiz on TCP

three-way handshake." Output: Markdown with 5 questions and Answers section. (See appendix for sample.) Test 2 - Vague: Input: "Give me something to study about networks." Agent asks: "Which networking topic do you want and what format?" Test 3 - Empty input: Agent asks for a topic and example format.

SECTION 6: REFLECTION 6.1 Hardest part: Separating responsibilities and designing JSON

schemas. 6.2 Enjoyed: Designing the Task Planner and pipeline. 6.3 Improvements: Implement Course Name: Growth OS AI Agent Development Challenge Student: Aaryaveer Yadav | Registration No: 23BAI10323 prototype, persistent preferences, Anki export. 6.4 Learned: Modularity and explicit formats improve chaining reliability. 6.5 Handling being stuck: Incremental debugging and adding validation steps.

SECTION 7: HACK VALUE Simulated session state tracking and modular micro-agent design to

allow chaining. Appendix A: Minimal orchestration pseudo-code (example) (See Python code file included in the ZIP for runnable implementation. API key placeholder is YOUR_API_KEY_HERE.) Appendix B: Prompt Variants (High School / College / Graduate) - High School Mode: simpler language, analogies, emojis, tip to explain to a friend. - College Mode: concise, technical, 1-3 sentence answers, "self-test tomorrow" suggestion. - Graduate Mode: conceptual reasoning, "Insight" lines, multi-sentence answers. Appendix C: Short API Setup Guide (no keys shown) 1. Go to https://platform.openai.com/ 2. Profile -> View API keys -> Create new secret key. 3. Copy it once and store securely. Use environment variables to keep it out of code. 4. In provided Python file, replace "YOUR_API_KEY_HERE" with your key on your local machine only. End of document.