

CSCI 420 Gen AI in Software Dev

Assignment 3 Report

Joshua Vasquez, Cathy Gao

1 Introduction:

With the advent and integration of AI, learning how to utilize different AI programs effectively has never been more necessary. Prompt engineering, though seemingly simple, can be a complicated task, requiring a lot of trial and error from the user to find the exact prompts which will get the model to give the best output. There are numerous prompting strategies which one can use on AI models, and we decided to use Few-Shot and Zero-Shot on GPT-4o-mini and Codestral-2501. The source code for our work can be found at https://github.com/cathieG/Prompt_Engineering.

2 Implementation:

2.1 Prompt Preparation and Models:

We first imported all the necessary libraries, utilizing our github token to access the AI models. prepared the prompts for Zero-Shot, taking the 22 different tasks, adding them to a list, then adding a simple prompt to them. We then gave these prompts as input into our two models (GPT-4o-mini and Codestral-2501). For each model across the different prompting techniques, we used a temperature of 0.3. Unfortunately, we found that the outputs were very long winded and not very good, forcing us to reevaluate our prompts. We found that by explicitly asking the models to only give us the final completed code or summary, we were able to reduce the amount of unneeded information by quite a bit, though some tasks always had the model returning some form of explanation. When we moved on to Few-Shot prompting, the process was exactly the same, though we did not need to rewrite the prompts as many times due to the experience we had from writing Zero-Shot prompts .

2.2 Testing and Evaluation:

We then calculated the “scores” of the output by comparing each model's output against each other. For natural language output, we used Bleu-4 score, and for code, we used exact match. All the prompts, outputs, evaluation scores, and our analysis of each task can be seen below. Afterwards, we put the information in a pandas dataframe, then converted it to a csv file.

2.3 Comparison and Comprehensive Analysis:

Overall, we were surprised to find that Few-Shot, while performing better than Zero-Shot, did not perform drastically better. In general, the code it produced was slightly more efficient avoiding the mistake of repeating code logic. We also found that, the more complex the coding tasks were, like tasks 21 and 22, the better Few-Shot performed relative to Zero-Shot. This makes sense, due to Few-Shot having examples to work off of. However, when it came to writing summaries (or anything revolving natural language), Zero-Shot was consistently better,

producing text that was more readable and concise. This could very much be a failing on our part due to the examples we provided in the prompts. However, we do not believe that to be the case, and found that the prompts from Zero-Shot were almost exactly what we were looking for. We thus conclude that Zero-shot is better for summarization or simple bug fixes while zero-shot is better for code completion or dealing with more complex code. In terms of the models, ChatGPT and Codestral were roughly equal. On average, ChatGPT seemed to perform better. However, while ChatGPT generally performed better, when Codestral performed better, it was usually by a bigger margin. This was most evident for task 11, where Codestral generated prompts across both prompting strategies that were much more detailed than ChatGPT's. It also showcased better coding for some tasks, such as task 12, where it accounted for negative values in a function, something ChatGPT did not do. On the other hand ChatGPT generally performed better with summarization or bug identification. Overall, for general or simple tasks, ChatGPT seemed to perform better. But for slightly more complicated, coding related tasks, Codestral seemed to provide better code. You can find analysis of each task below.

Task 1: Code Summarization (Java)

Models are asked to summarize the functionality of a Java method.

	Prompt
Zero-shot	Summarize what this function does in two sentences or less:
Few-shot	<p>You are a helpful assistant that summarizes the functionality of Java codes.</p> <pre>### Example 1 Code: public int add(int a, int b){ return a + b; } Summary: Returns the sum of two integers. ### Example 2 Code: public boolean isEven(int number) { return number % 2 == 0; } Summary: Checks whether a number is even. ### Now summarize the following method: Code: Prompt: Please provide a two-sentence or less summary of the following method: public Map < String , Integer > countWordFrequency (List < String > words) { Map < String , Integer > freqMap = new HashMap < > (); for (String word : words) { freqMap . put (word , freqMap . getOrDefault (word , 0) + 1); } return freqMap ; } Summary:</pre>

Task 3: Bug Classification (C++)

Models are asked to classify the type of bug in C++ functions:

	Prompt
Zero-shot	Classify the type of bug in the C++ function. Do nothing else besides this. This should be one sentence
Few-shot	<p>Your job is to classify the type of bug C++ functions.</p> <pre>### Example 1: code: int* getArray(int size) { int arr[size]; // Warning: local array return arr; // Bug: returning pointer to local variable) Bug Classification: Returning pointer to local variable Detailed Bug Explanation: The function attempts to return an array of integers. However, the array is declared as a local variable inside the function. Returning a pointer to this local array is invalid because the memory is deallocated once the function exits, leading to undefined behavior. ### Example 2: code: void printArray(int* arr, int size) { for (int i = 0; i <= size; ++i) { std::cout << arr[i] << std::endl; } } Bug Classification: Off-by-one error (out-of-bounds access) Detailed Bug Explanation: This function uses a loop to iterate through an array, but it goes one step beyond the last valid index. Since valid indices go from 0 to size - 1, accessing index size causes undefined behavior. Now, given the code int * getArray (int size) { int arr [size]; // Warning: local array return arr ; // Bug: returning pointer to local variable } Classify the type of bug in the C++ function. You only need to return the bug classification.</pre>

Task 5: Generating Flask APIs (Python)

Models are asked to create a '/greet/<username>' endpoint that returns a JSON greeting.

	Prompt
Zero-shot	Create a "/greet/<username>" endpoint that returns a JSON greeting. Don't explain your process. Just give me the completed code.
Few-shot	<p>You job is to create a "/greet/<username>" endpoint that returns a JSON greeting:</p> <pre>### Example 1: Original code: from flask import Flask, jsonify app = Flask(__name__) @app.route("/greet/<username>") def greet(username): # TODO: Return a JSON greeting pass Completed code: from flask import Flask, jsonify app = Flask(__name__) @app.route("/greet/<username>") def greet(username): return jsonify({"message": f"Hello, {username}!"}) ### Example 2: Original code: from flask import Flask, jsonify app = Flask(__name__) @app.route("/greet/<username>") def greet(username): # TODO: Return a JSON greeting pass Completed code: from flask import Flask, jsonify app = Flask(__name__) @app.route("/greet/<username>") def greet(username): response = {"message": f"Hello, " + username + "!"} return jsonify(response) Now, create a "/greet/<username>" endpoint that returns a JSON greeting given the code. No need for explanations or examples: from flask import Flask, jsonify app = Flask(__name__) @app . route ("/greet / < username > â€œ") def greet (username) : # TODO : Return a JSON greeting pass</pre>

Task 7: Null Dereference Detection (Java)

Models are asked to identify any null dereference risk in a function.

	Prompt
Zero-shot	Identify any null dereference risks in the code and summarize them briefly in a sentence or two.
Few-shot	<p>Your job is to identify potential null dereference bugs in Java code. These bugs happen when a method is called on an object that might be null, which can lead to a NullPointerException at runtime.</p> <pre>### Example 1: Original code: public String getUpperCase(String input) { return input.toUpperCase(); // Potential null dereference } Issue: The null dereference may occur on "line 2", where "input.toUpperCase()" is called without checking if "input" is null. ### Example 2: Original code: public int getMaxLength(User user) { return user.getName().length(); // Potential null dereference } Issue: A null dereference may occur on "line 2", where "user.getName()" could be null, and ".length()" is called on it without validation. Now, identify any potential null dereference issue in the following code. You only need to provide the issues. No need for explanations or examples: public int getLength (String s) { return s . length () ; // What if s is null ? }</pre>

Task 2: Bug Fixing (Python – Off-by-One)

Models are asked to identify and fix the off-by-one error in Python function.

	Prompt
Zero-shot	Identify and fix the off-by one error. Please provide only the completed function
Few-shot	<p>Your job is to identify the one-off bugs in python codes and fix them.</p> <pre>### Example 1 The function calculates the sum of all integers from start to end, inclusive. However, the original function uses range(start, end), which excludes the end value. This leads to an incomplete summation by omitting the final number. Original Code: def sum_range(start, end): total = 0 for i in range(start, end): total += i return total Corrected Code: def sum_range(start, end): total = 0 for i in range(start, end + 1): total += i return total The function is intended to add up all the integers from start to end, including both endpoints. But because range(start, end) doesn't include end, the function misses the last number and gives a result that's too small. Original Code: def sum_range(start, end): total = 0 for i in range(start, end): total += i return total Corrected Code: def sum_range (start , end) : total = 0 for i in range (start , end + 1) : total += i return total Describe the function, identify the one-off bug, and correct the code like the examples. You only need to return the corrected code.</pre>

Task 4: Generating Email Validators (Python + Regex)

Models are asked to complete a function using regex to validate basic email addresses:

	Prompt
Zero-shot	Complete the function using regex to validate basic email addresses. Don't explain your process, just give me the completed code.
Few-shot	<p>You are to help complete a function using regex to validate email addresses.</p> <pre>### Example 1: def is_valid_email(email): import re pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\$" return re.match(pattern, email) is not None ### Example 2: def is_valid_email(email): import re pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\$" return re.match(pattern, email) is not None Now, complete the following function using regex to validate basic email addresses. You just need to provide the final function. No need for explanations or examples.. def is_valid_email (email) : # TODO : Complete using regex pass</pre>

Task 6: SQL Schema Design (SQL)

Models are asked to write the schema for a review app with users, books, and reviews.

	Prompt
Zero-shot	Write the schema for a review app with users, books, and reviews. Don't explain your process, just give me the end result.
Few-shot	<pre>### Example 1: Given: -- TODO : Design schema with appropriate keys and constraints -- Tables: customers (id, name), products (id, name), orders (id, customer_id, product_id, quantity) Completed task: CREATE TABLE users (user_id INTEGER PRIMARY KEY, username TEXT NOT NULL); CREATE TABLE movies (movie_id INTEGER PRIMARY KEY, title TEXT NOT NULL); CREATE TABLE ratings (rating_id INTEGER PRIMARY KEY, user_id INTEGER, movie_id INTEGER, score INTEGER CHECK(score BETWEEN 1 AND 5), FOREIGN KEY (user_id) REFERENCES users(user_id), FOREIGN KEY (movie_id) REFERENCES movies(movie_id)); Now, do the same for the following code. No need for explanations or examples: -- TODO : Design schema with appropriate keys and constraints -- Tables : users (id , name), books (id , title), reviews (id , user_id , book_id , rating) Completed task: CREATE TABLE customers (id INTEGER PRIMARY KEY, name TEXT NOT NULL); CREATE TABLE products (id INTEGER PRIMARY KEY, name TEXT NOT NULL); CREATE TABLE orders (id INTEGER PRIMARY KEY, customer_id INTEGER, product_id INTEGER, quantity INTEGER, FOREIGN KEY (customer_id) REFERENCES customers(id), FOREIGN KEY (product_id) REFERENCES products(id)); ### Example 2: Given: -- TODO : Design schema with appropriate keys and constraints -- Tables: users (user_id, username), movies (movie_id, title), ratings (rating_id, user_id, movie_id, score) Completed task: CREATE TABLE users (user_id INTEGER PRIMARY KEY, username TEXT NOT NULL); CREATE TABLE movies (movie_id INTEGER PRIMARY KEY, title TEXT NOT NULL); CREATE TABLE ratings (rating_id INTEGER PRIMARY KEY, user_id INTEGER, movie_id INTEGER, score INTEGER CHECK(score BETWEEN 1 AND 5), FOREIGN KEY (user_id) REFERENCES users(user_id), FOREIGN KEY (movie_id) REFERENCES movies(movie_id));</pre>

Task 8: CSV Parser Variants (Python)

Models are asked to improve the parser to support quoted fields.

	Prompt
Zero-shot	Improve the parser to support quoted fields. Don't explain your process, just give me the final code.
Few-shot	<p>Your job is to identify limitations in naive CSV line parsers and suggest improvements. Naively splitting on commas does not handle quoted fields correctly. Provide examples of this issue.</p> <pre>### Example 1: Original code: def parse_csv_line(line): return line.split(',') Issue: This approach fails for input like "Smith, John",25,"New York, NY" because it will incorrectly split inside quoted strings. Fields containing commas must be treated as a single value if enclosed in quotes. ### Example 2: Original code: def parse_csv_line(line): return [cell.strip() for cell in line.split(',')] Issue: Even though this adds whitespace trimming, it still cannot handle lines like: "Doe, Jane", "Engineer, Senior", "R&D", where some fields contain commas. It will split those fields incorrectly. Now, improve the code. You only need to provide me with the improved code. No need for explanations or examples: def parse_csv_line (line) : return line . split (â€œ,â€œ) # Incomplete : doesn't handle quoted fields</pre>

Task 9: Data Class to API Conversion (Kotlin)

Models are asked to convert the data class to a REST API using Ktor:

	Prompt		
Zero-shot	Convert the data class to a REST API using Ktor. Just give me the final code without explaining your process.		
Few-shot	<p>Your task is to convert a Kotlin data class into a simple REST API using Ktor. For each example, create GET and POST routes that handle objects of the data class, using in-memory storage.</p> <p>## Example 1:</p> <p>Original code:</p> <pre>data class User(val id: Int, val name: String) // TODO: Create GET and POST endpoints using Ktor</pre> <p>Completed code:</p> <pre>import io.ktor.application.* import io.ktor.response.* import io.ktor.request.* import io.ktor.routing.* import io.ktor.features.* import io.ktor.serialization.* import io.ktor.server.engine.* import io.ktor.server.netty.* val users = mutableListOf<User>() fun main() { embeddedServer(Netty, port = 8080) { install(ContentNegotiation) { json() } routing { get("/users") { call.respond(users) } post("/users") { val user = call.receive<User>() users.add(user) call.respond(user) } } }.start(wait = true) } }</pre> <p>## Example 2:</p> <p>Original code:</p> <pre>data class Book(val id: Int, val title: String, val author: String) // TODO: Create GET and POST endpoints using Ktor</pre> <p>Completed code:</p> <pre>import io.ktor.application.* import io.ktor.response.* import io.ktor.request.* import io.ktor.routing.* import io.ktor.features.* import io.ktor.serialization.* import io.ktor.server.engine.* import io.ktor.server.netty.* import io.ktor.server.cio.* val books = mutableListOf<Book>() fun main() { embeddedServer(Netty, port = 8080) { install(ContentNegotiation) { json() } routing { get("/books") { call.respond(books) } post("/books") { val book = call.receive<Book>() books.add(book) call.respond(book) } } }.start(wait = true) } }</pre> <p>Now, convert the following data class into a REST API using Ktor. You only need to provide the final code. No need for explanations or examples:</p> <pre>data class Product(val id: Int, val name: String, val price: Double) // TODO: Create GET and POST endpoints using Ktor</pre>	<pre>call.respond(users) post("/users") { val user = call.receive<User>() users.add(user) call.respond(user) } }.start(wait = true) }</pre> <p>## Example 2:</p> <p>Original code:</p> <pre>data class Book(val id: Int, val title: String, val author: String) // TODO: Create GET and POST endpoints using Ktor</pre> <p>Completed code:</p> <pre>import io.ktor.application.* import io.ktor.response.* import io.ktor.request.* import io.ktor.routing.* import io.ktor.features.* import io.ktor.serialization.* import io.ktor.server.engine.* import io.ktor.server.netty.* import io.ktor.server.cio.* val books = mutableListOf<Book>() fun main() { embeddedServer(Netty, port = 8080) { install(ContentNegotiation) { json() } routing { get("/books") { call.respond(books) } post("/books") { val book = call.receive<Book>() books.add(book) call.respond(book) } } }.start(wait = true) } }</pre> <p>Now, convert the following data class into a REST API using Ktor. You only need to provide the final code. No need for explanations or examples:</p> <pre>data class Product(val id: Int, val name: String, val price: Double) // TODO: Create GET and POST endpoints using Ktor</pre>	<pre>install(ContentNegotiation) { json() } routing { get("/books") { call.respond(books) } post("/books") { val book = call.receive<Book>() books.add(book) call.respond(book) } } }.start(wait = true) }</pre> <p>Now, convert the following data class into a REST API using Ktor. You only need to provide the final code. No need for explanations or examples:</p> <pre>data class Product(val id: Int, val name: String, val price: Double) // TODO: Create GET and POST endpoints using Ktor</pre>

Task 11: Prompt from Code Comments (Python)

Models are asked to write a prompt (reverse prompt engineering) that could generate the provided code.

	Prompt	
Zero-shot	"Write a prompt that could generate the code below. No need to explain your reasoning, you can just give me the prompt:"	
Few-shot	<p>You are given a Python function with comments. Write a natural language prompt that could generate this code.</p> <p>## Example 1:</p> <p>Code:</p> <pre># This function returns the factorial of a number def factorial(n): if n == 0: return 1 return n * factorial(n - 1) Prompt: Write a recursive Python function to compute the factorial of a number.<p>## Example 2:</p><p>Code:</p><pre># This function returns the nth Fibonacci number using recursion def fibonacci(n):</pre></pre>	<pre>if n <= 1: return n return fibonacci(n - 1) + fibonacci(n - 2) Prompt: Create a Python function that returns the nth Fibonacci number using recursion. Now, write a prompt that would result in the generation of the following code. No need to explain your reasoning or give examples: # This function checks if a number is prime def is_prime(n): if n <= 1: return False for i in range(2, int(n**0.5) + 1): if n % i == 0: return False return True</pre>

Task 13: Linked List Node Deletion (C)

Models are asked to implement node deletion by value in provided codes.

	Prompt		
Zero-shot	implement node deletion by value. Just give me the finished code, don't explain anything.		
Few-shot	<p>You are given the definition of a singly linked list in C. Your task is to implement a function that deletes the first node with a given value from the list. Handle edge cases like deleting the head node or deleting a value that's not in the list.</p> <p>## Example 1:</p> <p>Given:</p> <pre>struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key) { // TODO: Implement node deletion }</pre> <p>Completed:</p> <pre>void deleteNode(struct Node** head, int key) { struct Node* temp = *head; struct Node* prev = NULL; // If head node itself holds the key if (temp != NULL && temp->data == key) { *head = temp->next; free(temp); return; } while (temp != NULL && temp->data != key) { prev = temp; temp = temp->next; } if (temp == NULL) return; // Unlink the node prev->next = temp->next; free(temp); }</pre> <p>## Example 2:</p> <p>Given:</p> <pre>struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key) { // TODO: Implement logic to remove node by value }</pre> <p>Completed:</p> <pre>void deleteNode(struct Node** head, int key) { // TODO: Implement node deletion }</pre>	<pre>// Search for the key while (temp != NULL && temp->data != key) { prev = temp; temp = temp->next; } // If key was not found if (temp == NULL) return; // Unlink the node prev->next = temp->next; free(temp); ## Example 2: Given: struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key) { // TODO: Implement logic to remove node by value }</pre> <p>Completed:</p> <pre>void deleteNode(struct Node** head, int key) { // TODO: Implement node deletion }</pre>	<pre>if (*head == NULL) return; struct Node* curr = *head; struct Node* prev = NULL; while (curr != NULL) { if (curr->data == key) { if (prev == NULL) { // Deleting head node *head = curr->next; } else { prev->next = curr->next; } free(curr); return; } prev = curr; curr = curr->next; } Now, implement the node deletion logic in the following code. You only need to provide the completed code.</pre> <pre>struct Node { int data; struct Node* next; }; void deleteNode (struct Node** head , int key) { // TODO : Implement node deletion }</pre>

Task 15: Constructor Completion (Python)

Models are asked to complete the class constructor in the provided codes.

	Prompt	
Zero-shot	Complete the class constructor. Don't explain your process, just give me the finished constructor.	
Few-shot	<p>You are to complete class constructors in Python by initializing attributes using "self".</p> <p>## Example 1:</p> <pre>class Book: def __init__(self, title, author): self.title = title self.author = author</pre> <p>This constructor initializes a 'Book' object with a title and an author.</p> <p>## Example 2:</p> <pre>class Student: def __init__(self, name, grade, email=None): self.name = name self.grade = grade self.email = email</pre> <p>This constructor initializes a 'Student' with name and grade, and an optional email (defaulting to None).</p> <p>Now, complete the class constructor so it takes 'name', 'age', and, optional 'email'. No need for explanations or examples:</p> <pre>class Person: def __init__(self): # TODO: Add name, age, and optional email pass</pre>	

Task 10: Function Summarization (Python)

Models are asked to write a brief summary of the function provided.

	Prompt	
Zero-shot	Summarize the function in two sentences or less:	
Few-shot	<p>You are given a Python function. Write a brief summary of what the function does.</p> <p>## Example 1:</p> <p>Function:</p> <pre>def square_elements(nums): return [x ** 2 for x in nums]</pre> <p>Summary:</p> <p>This function takes a list of numbers and returns a new list containing the squares of each number.</p> <p>## Example 2:</p> <p>Function:</p> <pre>def is_palindrome(s): return s == s[::-1]</pre> <p>Summary:</p> <p>This function checks whether a string is a palindrome by comparing it to its reverse.</p> <p>## Example 3:</p> <p>Function:</p> <pre>def count_vowels(s): return sum(1 for c in s.lower() if c in 'aeiou')</pre> <p>Summary:</p> <p>This function counts the number of vowels in a given string.</p> <p>Now, summarize the following function in two sentences or less:</p> <p>Function:</p> <pre>def reverse_words(sentence): return " ".join(sentence.split()[::-1])</pre>	

Task 12: Fixing Factorial Bug (Python)

Models are asked to fix the bug when input is 0:

	Prompt	
Zero-shot	Fix the bug when the input is 0. Just give me the final code without explaining anything else.	
Few-shot	<p>You are given a Python function that tries to compute the factorial of a number. If there's a bug when the input is 0, identify the issue and correct the function.</p> <p>## Example 1:</p> <p>Original Code:</p> <pre>def factorial(n): result = 1 for i in range(1, n): result *= i return result Bug: This function fails when n = 0. The factorial of 0 should be 1, but this code returns 1 without any iteration, which may seem fine, but it also gives incorrect results for n = 1 because it doesn't include n itself in the loop.<p>Corrected Code:</p><pre>def factorial(n): result = 1 for i in range(1, n + 1): result *= i return result</pre><p>## Example 2:</p><p>Original Code:</p><pre>def factorial(n): if n == 0: return 0</pre></pre>	<pre>result = 1 for i in range(1, n + 1): result *= i return result Bug: The function incorrectly returns 0 for n = 0. Mathematically, 0! is 1. Corrected Code: def factorial(n): if n == 0: return 1 for i in range(1, n + 1): result *= i return result ## Example 3: Now, fix the bug in the following code if the input is 0. You only need to provide me with the corrected code. No need for explanation or examples: def factorial(n): result = 1 for i in range(1, n): result *= i return result</pre>

Task 14: Recursive Function Completion (Python)

Models are asked to complete the recursive function for Fibonacci.

	Prompt	
Zero-shot	Complete the recursive function for Fibonacci. Only give me the completed code. Don't explain anything.	
Few-shot	<p>You are to complete recursive functions. Each function should include a base case and a recursive step.</p> <p>## Example 1:</p> <pre>def factorial(n): if n == 0 or n == 1: return 1 return n * factorial(n - 1)</pre> <p>This function returns the factorial of n using recursion. It stops when n is 0 or 1.</p> <p>## Example 2:</p> <pre>def sum_array(arr): if not arr: return 0 return arr[0] + sum_array(arr[1:])</pre> <p>This function returns the sum of all numbers in the list using recursion. It stops when the list is empty.</p> <p>Now, complete the recursive function to compute the nth Fibonacci number. You only need to provide the completed function. No need for explanations or examples:</p> <pre>def fibonacci(n): # TODO</pre>	

Task 16: Binary Search Completion (Java)

Models are asked to complete the binary search implementation in the provided codes.

	Prompt	
Zero-shot	Complete the binary search implementation and return only the final code without explanation.	
Few-shot	<p>You are to complete the binary search implementation by comparing the mid value to the target and adjusting the bounds accordingly.</p> <p>## Example 1:</p> <pre>public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = (left + right) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre> <p>This function implements a binary search to find the index of the target in a sorted array. It adjusts the left and right bounds based on comparisons with the middle value.</p> <p>## Example 2:</p> <pre>public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = (left + right) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre>	<pre>) else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } return -1; }</pre> <p>This binary search function also looks for the target in a sorted array, adjusting the search bounds based on comparisons with the middle element.</p> <p>Now, complete the binary search implementation by filling in the missing steps. You only need to provide me with the completed constructor. No need for explanations or examples:</p> <pre>public int binarySearch (int[] arr , int target) { int left = 0 , right = arr . length - 1 ; while (left <= right) { int mid = (left + right) / 2 ; // TODO : Compare and adjust bounds } return -1 ; }</pre>

Task 17: Self-Consistency Bug Fixing (C++)

Models are asked to resolve inconsistency between function name and logic in provided codes.

	Prompt	
Zero-shot	Resolve inconsistency between function name and logic. Don't explain what you did and why, just give me the corrected code.	
Few-shot	<p>Your task is to identify and resolve inconsistencies between function names and their logic.</p> <p>### Example 1:</p> <p>Original code:</p> <pre>// Supposed to return true if x is odd bool isEven(int x) { return x % 2 != 0; // Logic contradicts function name }</pre> <p>Completed task:</p> <pre>// Supposed to return true if x is odd bool isOdd(int x) { return x % 2 != 0; // Fixed: Corrected function name and logic }</pre> <p>### Example 2:</p> <p>Original code:</p> <pre>// Supposed to return true if x is positive bool isNegative(int x) { return x > 0; // Logic contradicts function name }</pre> <p>Completed task:</p> <pre>// Supposed to return true if x is positive bool isPositive(int x) { return x > 0; // Fixed: Corrected function name and logic }</pre>	<p>bool isPositive(int x) { returns result }</p> <p>Now, resolve the inconsistency in the following code. You only need to provide me the completed code. No need for explanation or examples:</p> <p>Code:</p> <pre>// Supposed to return true if x is even bool isOdd(int x) { return x % 2 == 0; // Logic contradicts function name }</pre>

Task 19: Summary Decomposition (C++)

Models are asked to decompose the high-level comment/summary into logical steps in the provided codes.

	Prompt	
Zero-shot	Take the summary/comment, decompose it, then rewrite it into more logical steps. Don't explain your process, just give me your final output.	
Few-shot	<p>Your task is to decompose the high-level comment into logical steps for the given code.</p> <p>### Example 1:</p> <p>Original code:</p> <pre>// Function that validates an input, calculates square, and returns result int process(int x) { if (x < 0) return -1; return x * x; }</pre> <p>Decomposed steps:</p> <ol style="list-style-type: none">1. Validate the input 'x' to ensure it is non-negative.2. If 'x' is negative, return '-1' to indicate an error or invalid input.3. If 'x' is non-negative, calculate the square of 'x' (i.e., 'x * x').4. Return the result of the square calculation. <p>### Example 2:</p> <p>Original code:</p> <pre>// Function that checks if a number is positive and returns a message string checkNumber(int x) {</pre>	<p>if (x > 0) return "Positive"; return "Not Positive"; }</p> <p>Decomposed steps:</p> <ol style="list-style-type: none">1. Check if the input number 'x' is greater than zero.2. If 'x' is positive, return the string "Positive".3. If 'x' is not positive (i.e., zero or negative), return the string "Not Positive". <p>Now, decompose the following function into logical steps based on its summary. Make it concise. No need to explain your reasoning or give examples:</p> <pre>// Function that validates an input, calculates square, and returns result int process(int x) { if (x < 0) return -1; return x * x; }</pre>

Task 21: Full-File Bug Detection and Refactoring (Python)

Models are asked to analyze the provided utility script. They are to identify any potential logic issues or design flaws, and then refactor it for better readability, correctness, and safety.

	Prompt		
	Analyze the utility script, find any logical issues or design flaws, then refactor the code for better readability, correctness, and safety. Don't explain your process or reasoning, just give me the refactored code with no comments.		
Few-shot	<p>Your task is to analyze the utility script below for any potential logic issues, design flaws, or readability concerns. Then, refactor the code to improve correctness, safety, and readability.</p> <p>### Example 1:</p> <p>Original code:</p> <pre>import csv def read_csv(filepath): with open(filepath, 'r') as f: return [row for row in csv.reader(f)] def summarize_column(data, index): values = [float(row[index]) for row in data[1:]] if len(values) == 0: return 0 total = sum(values) avg = total / len(values) return total, avg def main(): filepath = "data.csv" data = read_csv(filepath) total, avg = summarize_column(data, 1) print("Total:", total) print("Average:", avg) if __name__ == "__main__": main()</pre> <p>Refactored code:</p> <pre>import csv def read_csv(filepath): """Reads a CSV file and returns a list of rows.""" try: with open(filepath, 'r') as f: return [row for row in csv.reader(f)] except FileNotFoundError: print("Error: The file at {filepath} does not exist.") return []</pre>	<p>def summarize_column(data, index): """Summarizes a column in the CSV data, calculating the total and average.""" if not data: return 0, 0 values = [float(row[index]) for row in data[1:]] if len(values) == 0: return 0, 0 total = sum(values) avg = total / len(values) return total, avg def main(): """Main function to process CSV file and print summary.""" filepath = "data.csv" data = read_csv(filepath) if data: total, avg = summarize_column(data, 1) print("Total:", total) print("Average:", avg) else: print("No data to process.") if __name__ == "__main__": main()<p>Explanation: The original script had a few issues:</p><ul style="list-style-type: none">- If the file doesn't exist, the program would crash. We added exception handling for 'FileNotFoundError'.- If the 'data' list is empty or the column to summarize has no valid values, the program might divide by zero or fail. We added checks to return default values of '0' for both total and average in such cases.</p>	<p>- Added docstrings to functions for better clarity. - Included checks to ensure the script gracefully handles errors and edge cases.</p> <p>### Example 2:</p> <p>Original code:</p> <pre>def calculate_area(radius): return 3.14 * radius ** 2 def calculate_circumference(radius): return 2 * 3.14 * radius def main(): r = float(input("Enter the radius: ")) print("Area:", calculate_area(r)) print("Circumference:", calculate_circumference(r))</pre> <p>Refactored code:</p> <pre>import math def calculate_area(radius): """Calculates area of a circle given radius.""" return math.pi * radius ** 2 def calculate_circumference(radius): """Calculates circumference of a circle given radius.""" return 2 * math.pi * radius def main(): r = float(input("Enter the radius: ")) print("Area:", calculate_area(r)) print("Circumference:", calculate_circumference(r)) except ValueError: print("Error: Please enter a valid numeric value for radius.") if __name__ == "__main__": main()<p>Now, analyze and refactor the following script for improvements. You only need to provide the final refactored code, no need for explanation or examples:</p><p>Code</p><pre># utils.py - Script to parse and summarize numeric CSV files import csv def read_csv(filepath): with open(filepath, 'r') as f: return [row for row in csv.reader(f)] def summarize_column(data, index): values = [float(row[index]) for row in data[1:]] if len(values) == 0: return 0, 0 total = sum(values) avg = total / len(values) return total, avg def main(): filepath = "data.csv" data = read_csv(filepath) total, avg = summarize_column(data, 1) print("Total:", total) print("Average:", avg)</pre></pre>

Task 18: Prompt Chaining: Bug Identification → Fix (JavaScript)

Models are asked to identify and fix the bug in the provided codes.

	Prompt	
Zero-shot	Fix the bug in the function. Just give me the final code, and don't explain or write about anything else.	
Few-shot	<p>Your task is to identify the bug in the JavaScript function and fix it.</p> <p>### Example 1:</p> <p>Original code:</p> <pre>function isEven(n) { return n % 2; // Returns 1 or 0, not true/false }</pre> <p>Bug identification: The function returns '1' or '0', but it should return 'true' or 'false' for even or odd numbers.</p> <p>Fix:</p> <pre>function isEven(n) { return n % 2 === 0; // Corrected: Return true if even, false if odd }</pre> <p>Explanation: The original code returned '1' or '0', which is incorrect for an even/odd check. The fix uses strict equality check ('=== 0') to return 'true' for even numbers and 'false' for odd numbers.</p> <p>### Example 2:</p> <p>Original code:</p> <pre>function isPositive(n) { return n > 0 ? 1 : 'negative'; // Returns 1 or a string, not true/false }</pre>	<p>} Bug identification: The function returns '1' for positive numbers and 'negative' for non-positive numbers, but it should return 'true' or 'false'.</p> <p>Fix:</p> <pre>function isPositive(n) { return n > 0; // Corrected: Return true if positive, false otherwise }</pre> <p>Explanation: The function should return 'true' for positive numbers and 'false' for non-positive numbers. The original code returned '1' or a string, which doesn't match the intended behavior.</p> <p>Now, identify the bug and fix the following code:</p> <pre>function isEven (n) { return n % 2; // Returns 1 or 0, not true / false }</pre>

Task 20: Purpose Inference → Completion (Python)

Models are asked to complete the function provided based on intent.

	Prompt	
Zero-shot	Finish the rest of the function. Just give me the final function, nothing else:	
Few-shot	<p>Your task is to complete the function based on its intent.</p> <p>### Example 1:</p> <p>Original code:</p> <pre>def calculate_average(scores): total = 0 # TODO: Complete to return average pass</pre> <p>Completed function:</p> <pre>def calculate_average(scores): total = sum(scores) # Sum all scores return total / len(scores) if scores else 0 # Return the average</pre> <p>### Example 2:</p> <p>Original code:</p> <pre>def find_max(numbers): # TODO: Complete to return the maximum number pass</pre> <p>Completed function:</p> <pre>def find_max(numbers): max_num = max(numbers) # Use max function to get the maximum</pre>	<p>return max_num if numbers else None # Return the maximum or None if the list is empty</p> <p>Now, complete the following function based on its intent. You only need to provide the completed function. No need for explanations or examples:</p> <p>Code</p> <pre>def calculate_average (scores) : total = 0 # TODO : Complete to return average pass</pre>

Task 22: Code Completion and Robustness Enhancement (Python)

Models are asked to complete the provided file-processing script. The goal is to clean each line, remove punctuation, and count word frequencies correctly.

	Prompt		
Zero-shot	Complete the following code, making sure you write in the missing code as well as cleaning up each line, removing punctuation, and making sure it counts words frequencies correctly. Don't explain your process. Just output the completed code.		
Few-shot	<p>Your task is to complete and improve the following Python script that processes a text file. The script should clean each line by removing punctuation, converting the text to lowercase, and then count the frequency of each word in the file. Ensure robustness by handling edge cases.</p> <p>### Example 1:</p> <p>Original script:</p> <pre>import string def clean_line(line): # TODO: Remove punctuation and make lowercase pass def count_word_frequencies(filepath): word_counts = {} with open(filepath, 'r') as f: for line in f: clean = clean_line(line) word_count = count_word_frequencies(clean) word_counts[word] += 1 return word_counts</pre> <p>Refactored code:</p> <pre>def main(): filepath = "input.txt" lines = load_file(filepath) counts = count_word_frequencies(lines) print(word_counts)</pre>	<pre>import string def clean_line(line): # TODO: Remove punctuation and make lowercase pass def count_word_frequencies(filepath): word_counts = {} with open(filepath, 'r') as f: for line in f: clean = clean_line(line) word_count = count_word_frequencies(clean) word_counts[word] += 1 return word_counts</pre> <p>### Example 2:</p> <p>Original script:</p> <pre>def main(): filepath = "input.txt" lines = load_file(filepath) counts = count_word_frequencies(lines) print(word_counts)</pre>	<pre>def load_file(filepath): """Reads file and returns the lines.""" with open(filepath, 'r') as f: return f.readlines() def main(): """Processes the input file and prints word frequencies.""" filepath = "input.txt" lines = load_file(filepath) counts = count_word_frequencies(lines) print(word_counts)</pre> <p>Now, analyze and complete the following script to process a file and count word frequencies. You only need to provide the final script. No need for explanations or examples:</p> <p>Code</p> <pre># file_processor.py - Incomplete script for processing text files import string def load_file(filepath): with open(filepath, 'r') as f: return f.readlines() def clean_line(line): # TODO: Remove punctuation and make lowercase pass def count_word_frequencies(filepath): word_counts = {} for line in lines: clean = clean_line(line) word_count = count_word_frequencies(clean) word_counts[word] += 1 return word_counts</pre>

Task 1: Code Summarization (Java)

Models are asked to summarize the functionality of a Java method.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	The function <code>countWordFrequency</code> takes a list of words as input and returns a map that counts the frequency of each word in the list. It uses a <code>HashMap</code> to store the words as keys and their corresponding counts as values, updating the count for each word as it iterates through the list.	This function takes a list of words and returns a map where the keys are the words from the list and the values are the frequencies of those words. It iterates through the list, updating the count for each word in the map.	Bleu-4 Score: 30.65076
Few-Shot	Summary: Counts the frequency of each word in a list and returns a map with words as keys and their corresponding counts as values.	The method <code>countWordFrequency</code> takes a list of strings and returns a map where each string is a key and its value is the frequency of that string in the list. It iterates through the list, updating the frequency count for each word in the map.	Bleu-4 Score: 9.4884
Analysis:	The summaries for both models and both prompting strategies were very similar, both correctly describing the function. In fact, prompting strategy made more of a difference than the models used. Interestingly, the two best summaries are GPT with Zero-Shot and Codestral with Few-Shot (giving the most detailed answers), indicating that for summaries, those strategies may be better for those models. This was also the easiest task to get only the desired output (only the summary, no explanations). Overall, Zero-Shot seems better for summaries to due the lack of need for examples.		

Task 3: Bug Classification (C++)

Models are asked to classify the type of bug in C++ functions:

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	The bug is returning a pointer to a local variable, which leads to undefined behavior.	Returning pointer to local variable	Bleu-4 Score: 2.74093
Few-Shot	Bug Classification: Returning pointer to local variable	Returning pointer to local variable	Bleu-score: 54.88116
Analysis:	Both prompting strategies provided (meaningfully) the same, correct bug classifications (returning pointer to local variable). For both prompt styles, Chat GPT provided the more in-depth response, especially for Zero-Shot, where it provided an explanation of why it is a bug. Similar to task 2, simple bug classification seems to only require Zero-Shot prompting, buugs may require Few-Shot prompting.		

Task 5: Generating Flask APIs (Python)

Models are asked to create a `/greet/<username>` endpoint that returns a JSON greeting.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>python from flask import Flask, jsonify app = Flask(__name__) @app.route('/greet/<username>') def greet(username): return jsonify(greeting=f"Hello, {username}!") if __name__ == '__main__': app.run(debug=True)</pre>	<pre>python from flask import Flask, jsonify app = Flask(__name__) @app.route('/greet/<username>') def greet(username): return jsonify(message=f"Hello, {username}!") if __name__ == '__main__': app.run(debug=True)</pre>	Exact Match: False
Few-Shot	<pre>python from flask import Flask, jsonify app = Flask(__name__) @app.route('/greet/<username>') def greet(username): return jsonify({'message': f'Hello, {username}!'})</pre>	<pre>python from flask import Flask, jsonify app = Flask(__name__) @app.route('/greet/<username>') def greet(username): return jsonify({'message': f'Hello, {username}!'})</pre>	Exact Match: True
Analysis:	The functionality across all four is virtually the same, returning a <code>Json</code> greeting. However, Zero-Shot provided a <code>__main__</code> block, which allows the file to be run directly. This is certainly helpful, but was not required. If users are unfamiliar with <code>Json</code> , then Zero-Shot prompting may be helpful in providing them with code they would not have thought of. And as all functions are nearly identical, there seems no need to leverage the extra examples of Few-Shot prompting.		

Task 2: Bug Fixing (Python – Off-by-One)

Models are asked to identify and fix the off-by-one error in Python function.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>python def sum_range(start, end): total = 0 for i in range(start, end + 1): total += i return total</pre>	<pre>python def sum_range(start, end): total = 0 for i in range(start, end + 1): total += i return total</pre>	Exact Match: True
Few-Shot	<pre>python def sum_range(start, end): total = 0 for i in range(start, end + 1): total += i return total</pre>	<pre>python def sum_range(start, end): total = 0 for i in range(start, end + 1): total += i return total</pre>	Exact Match: False
Analysis:	Prompting strategy and model had no effect on the output, indicating that for simple bug fixes, each model knows how to perform very well. Each function is correct, adding plus 1 to "end", making all the functions correct. For simple bug fixes, Zero-Shot prompting seems adequate and better, though more complex problems may require more detailed prompts such as Few-Shot.		

Task 4: Generating Email Validators (Python + Regex)

Models are asked to complete a function using regex to validate basic email addresses:

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>python import re def is_valid_email(email): pattern = r'^[a-zA-Z0-9_%.+]*@[a-zA-Z0-9-]+\.[a-zA-Z]{2,}\$' return re.match(pattern, email) is not None</pre>	<pre>python import re def is_valid_email(email): pattern = r'^[a-zA-Z0-9_%.+]*@[a-zA-Z0-9-]+\.[a-zA-Z]{2,}\$' return bool(re.match(pattern, email))</pre>	Exact Match: False
Few-Shot	<pre>python def is_valid_email(email): import re pattern = r'^[w\.-]+@[w\.-]+\w+\$' return re.match(pattern, email) is not None</pre>	<pre>python def is_valid_email(email): import re pattern = r'^[a-zA-Z0-9_%.+]*@[a-zA-Z0-9-]+\.[a-zA-Z]{2,}\$' return re.match(pattern, email) is not None</pre>	Exact Match: False
Analysis:	Both Zero-Shot outputs included import statements which were not in Few-Shot, meaning zero shot performed better. The outputs are virtually identical across all four. Codestral Zero-Shot converts <code>re.match</code> into a <code>bool</code> first, which is different from checking if a match was found, but they are functionally similar. For code completion of simple functions, Zero-Shot prompting seems superior, as not complete examples could cause oversight in Few-Shot prompting, such as missing import statements.		

Task 6: SQL Schema Design (SQL)

Models are asked to write the schema for a review app with users, books, and reviews.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>sql CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(100) NOT NULL); CREATE TABLE books (id SERIAL PRIMARY KEY, title VARCHAR(255) NOT NULL); CREATE TABLE reviews (id SERIAL PRIMARY KEY, user_id INT NOT NULL, book_id INT NOT NULL, rating INT CHECK (rating >= 1 AND rating <= 5), FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE, FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE);</pre>	<pre>sql CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(255) NOT NULL); CREATE TABLE books (id SERIAL PRIMARY KEY, title VARCHAR(255) NOT NULL); CREATE TABLE reviews (id SERIAL PRIMARY KEY, user_id INT NOT NULL, book_id INT NOT NULL, rating INT CHECK (rating >= 1 AND rating <= 5), FOREIGN KEY (user_id) REFERENCES users(id), FOREIGN KEY (book_id) REFERENCES books(id)</pre>	Exact Match: False
Few-Shot	<pre>sql CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT NOT NULL); CREATE TABLE books (id INTEGER PRIMARY KEY, title TEXT NOT NULL); CREATE TABLE reviews (id INTEGER PRIMARY KEY, user_id INTEGER, book_id INTEGER, rating INTEGER CHECK(rating BETWEEN 1 AND 5), FOREIGN KEY (user_id) REFERENCES users(id), FOREIGN KEY (book_id) REFERENCES books(id)</pre>	<pre>sql CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT NOT NULL); CREATE TABLE books (id INTEGER PRIMARY KEY, title TEXT NOT NULL); CREATE TABLE reviews (id INTEGER PRIMARY KEY, user_id INTEGER, book_id INTEGER, rating INTEGER CHECK(rating BETWEEN 1 AND 5), FOREIGN KEY (user_id) REFERENCES users(id), FOREIGN KEY (book_id) REFERENCES books(id)</pre>	Exact Match: True
Analysis:	Few shot prompting created the same schema across models. The schema defines everything it need to, and uses integer for id, not auto-incrementing by default. Zero-Shot for chatGPT produced the same code as well. However, zero shot for codestral was different, using serial for ID instead of integers. Due to potential differences in schema output, prompt quality can affect the minor details about the code, which may be important. Due to this, adding examples can guide the model to producing schemas that align with what the user is trying to do better.		

Models are asked to identify any null dereference risk in a function.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	The code contains a null dereference risk because if the input string 's' is 'null', calling 's.length()' will result in a 'NullPointerException'. To mitigate this risk, a null check should be added before attempting to access the length of the string.	The code contains a null dereference risk. If the input string 's' is 'null', calling 's.length()' will result in a 'NullPointerException' because the 'length()' method cannot be invoked on a 'null' object. To mitigate this risk, you should add a null check before calling 's.length()'.	Bleu-4 Score: 51.54409
Few-Shot	Issue: A null dereference may occur on **line 2**, where 's.length()' is called without checking if 's' is null.	Potential null dereference: - Line 2: 's.length()' is called without checking if 's' is null.	Bleu-score: 52.73931
Analysis:	Few-Shot and Zero-Shot both identify the null dereference risk at the correct line. However, Zero-Shot provides a more in-depth explanation on how to mitigate the risk. In our opinion, Few-Shot provides the better identification of the null dereference risk, making the explanation more concise and readable. Between models, Codestral provided the more assertive output, saying something will happen versus chatGPT saying something might happen. Codestral is better in this regard, as ChatGPT not being certain could cause users to not think the risk needs fixing. As has been pretty standard so far, the prompting strategies don't seem to make a huge difference for identification of small problems, though prompting strategies that provide examples can allow the model to output develop more specific identifications.		

Models are asked to convert the data class to a REST API using Ktor:

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre> -- kotlin import io.ktor.application.* import io.ktor.features.CorsConfiguration import io.ktor.routing.* @ktor http { install(ContentNegotiation) install(Cors) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } fun main() { embedderServer(Netty, port = 8080) { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } }</pre>	<pre> -- kotlin import io.ktor.application.* import io.ktor.features.CorsConfiguration import io.ktor.routing.* @ktor http { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } fun main() { embedderServer(Netty, port = 8080) { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } }</pre>	Exact Match: False
Few-Shot	<pre> -- kotlin import io.ktor.application.* import io.ktor.features.CorsConfiguration import io.ktor.routing.* @ktor http { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } fun main() { embedderServer(Netty, port = 8080) { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } }</pre>	<pre> -- kotlin import io.ktor.application.* import io.ktor.features.CorsConfiguration import io.ktor.routing.* @ktor http { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } fun main() { embedderServer(Netty, port = 8080) { install(ContentNegotiation) routing { get("/products") { val products = listOf(Product("Apple", "iPhone 15 Pro", 999), Product("Samsung", "Galaxy S24 Ultra", 899), Product("Google", "Pixel 8 Pro", 799), Product("Huawei", "Mate X2 Fold", 1499), Product("Xiaomi", "Mi Mix Fold", 1299)) call.respond(HttpStatusCode.OK, products) } } } }</pre>	Exact Match: True
Analysis:	<p>For ChatGPT, the biggest difference in the code is the use of Muxer in Zero-Shot. Because it uses Muxer, the server in Zero-Shot is safer to use, though less efficient than the one which Few-Shot provided. For Codestral, the Zero-Shot server tells the user that the product was stored while the Few-Shot server returns the product as a JSON. Overall, the code quality of Few-Shot seems better, being more applicable to the real-world and more computationally efficient across both models. While the ChatGPT Few-Shot is not as safe, this can be fixed with specific prompting. Both models have roughly similar execution.</p>		

Models are asked to write a prompt (reverse prompt engineering) that could generate the provided code.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	Write a Python function named <code>is_prime</code> that takes an integer <code>n</code> and is intended to check if the number is prime.	Write a Python function named <code>`is_prime`</code> that takes an integer <code>`n`</code> as input and returns <code>`True`</code> if <code>`n`</code> is a prime number, and <code>`False`</code> otherwise.	Bleu-4 Score: 18.14559
Few-Shot	Write a Python function that checks if a number is prime.	Write a Python function that checks if a given number is prime. The function should return <code>`True`</code> if the number is prime and <code>`False`</code> otherwise.	Bleu-score: 29.55261
Analysis:	Codestral performed better on this task than GPT, providing a more in-depth prompt that has a better chance of generating the desired code through more specific details. Oddly, the Zero-Shot prompts were more detailed than the Few-Shot prompts, which doesn't make a lot of sense to us. However, this could be an issue with the examples we provided the model. In terms of prompt generation, it seems like Zero-Shot outperforms Few-Shot for general purposes, though more detailed example would certainly lead to more detailed prompts from the model.		

Models are asked to improve the parser to support quoted fields.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre> python import csv def parse_csv_line(line): return next(csv.reader([line])) </pre>	<pre> python def parse_csv_line(line): fields = [] field = [] in_quotes = False escape_next = False for char in line: if escape_next: field.append(char) escape_next = False elif char == '"': escape_next = True elif char == ',' and not in_quotes: in_quotes = not in_quotes fields.append(''.join(field)) field = [] else: field.append(char) fields.append(''.join(field)) return fields </pre>	Exact Match: False
Few-Shot	<pre> python import csv from io import StringIO def parse_csv_line(line): return next(csv.reader(StringIO(line))) </pre>	<pre> python def parse_csv_line(line): import csv import io return next(csv.reader(io.StringIO(line))) </pre>	Exact Match: False
Analysis:	<p>ChatGPT produced similar code between prompting strategies, with the parser correctly handling quoted fields of csv files, though Few-Shot uses StringIO, which is odd since import statements were not used in the examples. On the other hand, Codestral demonstrated much better code using Few-Shot. When using Zero-Shot, though generating code that works, it manually parses through a CSV line by line. It is less computationally efficient and also more prone to errors, due to all the steps that could go wrong. As opposed to the first couple of tasks, task 8 showcases how prompting strategies can drastically affect code quality, showcasing that adding examples can help models produce better code. Overall, ChatGPT performed better for this task. And Few-Shot performed better.</p>		

Models are asked to write a brief summary of the function provided.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<p>The `reverse_words` function takes a string `sentence`, splits it into individual words, reverses the order of those words, and then joins them back together into a single string with spaces in between.</p> <p>The result is the original sentence with the word order reversed.</p>	<p>The `reverse_words` function takes a sentence as input and returns the sentence with the order of words reversed. It splits the sentence into words, reverses the list of words, and then joins them back into a single string.</p>	Bleu-4 Score: 39.6663
Few-Shot	<p>This function takes a sentence as input and returns a new sentence with the order of the words reversed. It splits the sentence into words, reverses the list of words, and then joins them back together with spaces.</p>	<p>This function takes a sentence as input and returns a new sentence with the order of the words reversed. It splits the input sentence into words, reverses the list of words, and then joins them back into a single string.</p>	Bleu-score: 82.74856
Analysis:	<p>Though there are minor differences between all four, the summaries all correctly explain what the function does. However, Zero-Shot provided the better summaries for two reasons. Firstly, though small, it included the function name, which is helpful. Secondly, it explains what the function does, how it does it, and then says the result. In comparison, Few-Shot explains what the function returns, then explains how it does it. Though this can be partly personal preference, the summaries from Zero-Shot make for an easier reading experience, and more closely resemble typical summaries. Zero-Shot prompting seems to be better for summarizing functions, though again, models may require more examples for more complex functions. The models performed about the same as well.</p>		

Models are asked to fix the bug when input is 0:

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre> python def factorial(n): if n == 0: return 1 result = 1 for i in range(1, n + 1): result *= i return result </pre>	<pre> python def factorial(n): if n == 0: return 1 result = 1 for i in range(1, n + 1): result *= i return result </pre>	Exact Match: True
Few-Shot	<pre> python def factorial(n): result = 1 for i in range(1, n + 1): result *= i return result </pre>	<pre> python def factorial(n): result = 1 for i in range(1, n + 1): result *= i return result </pre>	Exact Match: True
Analysis:	<p>Due to the simple nature of the bug fix, all models and prompts produced the same corrected function, allowing the function to take an input of zero. As stated for previous bug fixing tasks, Zero-Shot be enough to handle simple bug fixes, though models may need more examples for more complex code. Both models performed the same.</p>		

Task 13: Linked List Node Deletion (C)

Models are asked to implement node deletion by value in provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...C #include <stdio.h> struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key){ struct Node* temp = *head; struct Node* prev = NULL; // If the head node itself holds the key to be deleted if(temp->data==key){ *head = temp->next; // Changed head free(temp); // Free old head return; } // Search for the key to be deleted, keep track of the previous node while(temp != NULL && temp->data != key){ prev = temp; temp = temp->next; } // If the key was not present in the linked list if(temp == NULL) return; // Unlink the node from linked list prev->next = temp->next; free(temp); // Free memory }</pre>	<pre>...C struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key){ struct Node* temp = *head; struct Node* prev = NULL; if(*head == NULL) return; struct Node* curr = *head; struct Node* next = NULL; while(curr != NULL){ if(curr->data == key){ // Deleting head node *head = curr->next; free(curr); return; } prev = curr; curr = curr->next; } return;</pre>	Exact Match: False
Few-Shot	<pre>...C #include <stdio.h> struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key){ struct Node* temp = *head; struct Node* prev = NULL; while(temp != NULL){ if(temp->data == key){ // Deleting head node *head = temp->next; free(temp); return; } prev = temp; temp = temp->next; } return;</pre>	<pre>...C struct Node { int data; struct Node* next; }; void deleteNode(struct Node** head, int key){ struct Node* temp = *head; struct Node* prev = NULL; if(*head == NULL) return; struct Node* curr = *head; struct Node* next = NULL; while(curr != NULL){ if(curr->data == key){ // Deleting head node *head = curr->next; free(curr); return; } prev = curr; curr = curr->next; } return;</pre>	Exact Match: False
Analysis:	For GPT, the Zero-Shot code separates head handling before the while loop, while the Few-Shot version handles everything inside the While loop, making it slightly more compact and not repeating in logic. Codestral follows a similar pattern, with the Zero-Shot handling head deletion outside the main loop and the Few-Shot handling it inside the loop. Overall, due to the examples provided to Few-Shot, it was able to produce better, more efficient code.		

Task 15: Constructor Completion (Python)

Models are asked to complete the class constructor in the provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...python class Person: def __init__(self, name, age, email=None): self.name = name self.age = age self.email = email</pre>	<pre>...python class Person: def __init__(self, name, age, email=None): self.name = name self.age = age self.email = email</pre>	Exact Match: True
Few-Shot	<pre>...python class Person: def __init__(self, name, age, email=None): self.name = name self.age = age self.email = email This constructor initializes a `Person` object with a name, age, and an optional email (defaulting to `None`).</pre>	<pre>...python class Person: def __init__(self, name, age, email=None): self.name = name self.age = age self.email = email</pre>	Exact Match: False
Analysis:	All outputs are effectively the same, with the only difference being that ChatGPT provided an explanation of the constructor. This is slightly odd, as the prompt asked for no explanations. This issue was not presented with Codestral. All prompting strategies and models correctly completed the class constructor with the necessary constructor parameters. The simple nature of adding class parameters makes Zero-Shot a viable option, though needing specific parameters based on existing code may require the model to have examples.		

Task 17: Self-Consistency Bug Fixing (C++)

Models are asked to resolve inconsistency between function name and logic in the provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...cpp bool isEven(int x) { return x % 2 == 0; }</pre>	<pre>...cpp bool isEven(int x) { return x % 2 == 0; }</pre>	Exact Match: True
Few-Shot	<pre>...cpp // Supposed to return true if x is even bool isEven(int x) { return x % 2 == 0; // Fixed: Corrected function name and logic }</pre>	<pre>...cpp // Supposed to return true if x is even bool isEven(int x) { return x % 2 == 0; // Fixed: Corrected function name and logic }</pre>	Exact Match: True
Analysis:	Both models and prompting strategies identified the issue with the model name and function and corrected it, changing the name from isOdd to isEven. It's worth noting for Zero-Shot that it decided to change the function name instead of the function itself. While this seems obvious, due to the simplicity of the function, it is possible that it could have changed the function return statement instead, especially since we didn't specify how to fix the issue. For Few-Shot, it makes sense that it changed the name as that was how it was done in the examples. While Zero-Shot prompting can work, Few-Shot prompting/providing examples to the model will help the model make changes in ways that are user wants.		

Task 14: Recursive Function Completion (Python)

Models are asked to complete the recursive function for Fibonacci.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...python def fibonacci(n): if n <= 0: return 0 elif n == 1: return 1 else: return fibonacci(n - 1) + fibonacci(n - 2)</pre>	<pre>...python def fibonacci(n): if n <= 0: return 0 elif n == 1: return 1 else: return fibonacci(n-1) + fibonacci(n-2)</pre>	Exact Match: False
Few-Shot	<pre>...python def fibonacci(n): if n == 0: return 0 elif n == 1: return 1 return fibonacci(n - 1) + fibonacci(n - 2)</pre>	<pre>...python def fibonacci(n): if n <= 0: return 0 elif n == 1: return 1 return fibonacci(n - 1) + fibonacci(n - 2)</pre>	Exact Match: False
Analysis:	Perhaps due to Fibonacci being a common problem, the different prompts basically produced the same output, with the only real difference being in spacing. However, one big difference in the GPT outputs is tt the Zero-Shot function accounts for negative numbers, while the Few-Shot prompt does not, making the Zero-Shot code better. This oversight is not present in Codestral. This may indicate that, for coding problems in which there is a common, simple answer (such as Fibonacci), Few-Shot prompt could actually be hurt by examples, as those examples could steer the model away from the common answer. In this case, Zero-Shot prompting performed better.		

Task 16: Binary Search Completion (Java)

Models are asked to complete the binary search implementation in the provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = (left + right) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre>	<pre>...java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = left + (right - left) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre>	Exact Match: False
Few-Shot	<pre>...java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = (left + right) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre>	<pre>...java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = left + (right - left) / 2; // Avoid potential overflow if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }</pre>	Exact Match: False
Analysis:	Zero-Shot and Few-Shot provided the same Binary search implementation for each model. And while the prompts produce a function that works, there is a difference between the models. In Codestral, the function avoids integer overflow by calculating the distance between the two indices, whereas the function in GPT does not account for this. In this case, while the prompting strategies performed roughly the same, Codestral performed better than GPT on both strategies.		

Task 18: Prompt Chaining: Bug Identification + Fix (JavaScript)

Models are asked to identify and fix the bug in the provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>...javascript function isEven(n) { return n % 2 === 0; }</pre>	<pre>...javascript function isEven(n) { return n % 2 === 0; }</pre>	Exact Match: True
Few-Shot	<pre>...javascript function isEven(n) { return n % 2 === 0; // Corrected: Return true if even, false if odd }</pre>	<pre>...javascript function isEven(n) { return n % 2 === 0; // Corrected: Return true if even, false if odd }</pre>	Exact Match: False
Analysis:	In the Bug Identification and Fix task, GPT-4o mini and Codestral-2501 were given a JavaScript function to check if a number is even. Both models correctly recognized that the original code had no bugs and returned the code unchanged. In the Few-Shot setting, they added brief explanatory comments, clarifying the function's behavior, but slightly differed in format, so Exact Match was False. Overall, both models demonstrated solid understanding, but minor formatting choices affected the Few-Shot match.		

Task 19: Summary Decomposition (C++)

Models are asked to decompose the high-level comment/summary into logical steps in the provided codes.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>1. Define a function named `process` that takes an integer input `x`. 2. Check if `x` is less than 0: - If true, return -1. 3. If `x` is 0 or greater, calculate the square of `x` by multiplying `x` by itself. 4. Return the squared result.</pre>	<pre>'''c int process(int x){ if (x < 0){ return -1; } int result = x * x; return result; } '''</pre>	Bleu-4 Score: 1.34122
Few-Shot	<pre>1. Validate the input `x` to ensure it is non-negative. 2. If `x` is negative, return -1 to indicate an error or invalid input. 3. If `x` is non-negative, calculate the square of `x` (i.e., `x * x`). 4. Return the result of the square calculation.</pre>	<pre>1. Validate the input `x` to ensure it is non-negative. 2. If `x` is negative, return -1 to indicate an error or invalid input. 3. If `x` is non-negative, calculate the square of `x` (i.e., `x * x`). 4. Return the result of the square calculation.</pre>	Bleu-score: 100.0
Analysis:	<p>In the Decomposition task, GPT-4o mini and Codestral-2501 were asked to break down a simple C function into logical steps. In the Zero-Shot setting, GPT-4o mini captured the major operations correctly but used a more direct, slightly less polished list. With Few-Shot examples, both models produced detailed, structured decompositions that matched perfectly, leading to a Bleu-4 score of 100.0. Overall, Few-Shot examples helped both models fully align with the task's clarity and structure expectations.</p>		

Task 21: Full-File Bug Detection and Refactoring (Python)

Models are asked to analyze the provided utility script. They are to identify any potential logic issues or design flaws, and then refactor it for better readability, correctness, and safety.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>''' python import csv def read_csv(filepath): with open(filepath, 'r') as f: return [row for row in csv.reader(f)] def summarize_column(data, index): total = sum(data[index]) print('Total:', total) print('Average:', avg) else: print('No data to process.') return 0, 0 if __name__ == '__main__': main() '''</pre>	<pre>''' python # Utility.py - Script to parse and summarize numeric CSV files import csv def read_csv(filepath): with open(filepath, 'r') as f: return list(csv.reader(f)) def summarize_column(data, index): if not data or len(data) < 2: raise ValueError("Data must contain at least one header row and one data row.") values = [float(row[index]) for row in data[1:]] total = sum(values) avg = total / len(values) return total, avg '''</pre>	Exact Match: False
Few-Shot	<pre>''' python # Utility.py - Script to parse and summarize numeric CSV files import csv def read_csv(filepath): """Reads a CSV file and returns a list of rows.""" try: with open(filepath, 'r') as f: return [row for row in csv.reader(f)] except FileNotFoundError: print("Error: The file at (filepath) does not exist.") return [] except Exception as e: print(f"An error occurred while reading the file: {e}") return [] def summarize_column(data, index): """Summarizes a column in the CSV data, calculating the total and average.""" if not data or len(data) < 1: Check if data is empty or has no rows except header return 0, 0 try: values = [float(row[index]) for row in data[1:]] total = sum(values) avg = total / len(values) print('Total:', total) print('Average:', avg) else: print('No data to process.') return 0, 0 except: print('Error: Invalid index or data type.') return 0, 0 '''</pre>	<pre>''' python # Utility.py - Script to parse and summarize numeric CSV files import csv def read_csv(filepath): """Reads a CSV file and returns a list of rows.""" try: with open(filepath, 'r') as f: return [row for row in csv.reader(f)] except FileNotFoundError: print("Error: The file at (filepath) does not exist.") return [] except Exception as e: print(f"An error occurred while reading the file: {e}") return [] def summarize_column(data, index): """Summarizes a column in the CSV data, calculating the total and average.""" if not data or len(data) < 1: Check if data is empty or has no rows except header return 0, 0 try: values = [float(row[index]) for row in data[1:]] total = sum(values) avg = total / len(values) print('Total:', total) print('Average:', avg) else: print('No data to process.') return 0, 0 except: print('Error: Invalid index or data type.') return 0, 0 '''</pre>	Exact Match: False
Analysis:	<p>In the Full-File Bug Detection and Refactoring task, both GPT-4o mini and Codestral-2501 improved the original script. In the Zero-Shot setting, they made some structural refinements but missed important safety checks. With Few-Shot examples, both models made clearer improvements: they added docstrings, handled file errors, validated inputs, and made the main function more organized. Few-Shot guidance led to noticeably better results, with Codestral favoring a more cautious style and GPT-4o producing slightly more concise code.</p>		

Task 20: Purpose Inference + Completion (Python)

Models are asked to complete the function provided based on intent.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>''' python def calculate_average(scores): total = sum(scores) return total / len(scores) if scores else 0 '''</pre>	<pre>''' python def calculate_average(scores): total = 0 for score in scores: total += score return total / len(scores) '''</pre>	Exact Match: False
Few-Shot	<pre>''' python def calculate_average(scores): total = sum(scores) # Sum all scores return total / len(scores) if scores else 0 # Return the average '''</pre>	<pre>''' python def calculate_average(scores): total = sum(scores) # Sum all scores return total / len(scores) if scores else 0 # Return the average '''</pre>	Exact Match: True
Analysis:	<p>Both GPT-4o mini and Codestral-2501 successfully completed the calculate_average function. In the Zero-Shot setting, GPT-4o mini included a check to handle empty input, while Codestral initially did not. With Few-Shot examples, both models produced the same correct and safe solution, earning an exact match. Overall, Few-Shot guidance helped align their outputs fully with expectations.</p>		

Task 22: Code Completion and Robustness Enhancement (Python)

Models are asked to complete the provided file-processing script. The goal is to clean each line, remove punctuation, and count word frequencies correctly.

	GPT-4o mini	Codestral-2501	Evaluation Metrics:
Zero-Shot	<pre>''' python # file_processor.py - Incomplete script for processing text files import string def load_file(filepath): with open(filepath, 'r') as f: return f.readlines() def clean_line(line): return line.translate(str.maketrans('', '', string.punctuation)).lower() def count_words(lines): word_counts = {} for line in lines: clean = clean_line(line) words = clean.split() for word in words: word_counts[word] = word_counts.get(word, 0) + 1 return word_counts '''</pre>	<pre>''' python # file_processor.py - Incomplete script for processing text files import string def load_file(filepath): with open(filepath, 'r') as f: return f.readlines() def clean_line(line): return line.translate(str.maketrans('', '', string.punctuation)).lower() def count_words(lines): word_counts = {} for line in lines: clean = clean_line(line) words = clean.split() for word in words: word_counts[word] = word_counts.get(word, 0) + 1 return word_counts '''</pre>	Exact Match: False
Few-Shot	<pre>''' python # file_processor.py - Script for processing text files and counting word frequencies import string def load_file(filepath): """Reads and returns lines from the given file.""" with open(filepath, 'r') as f: return f.readlines() def clean_line(line): """Removes punctuation, converts text to lowercase, and returns the cleaned line.""" translator = str.maketrans('', '', string.punctuation) return line.translate(translator).lower() def count_words(lines): """Main function to process the file and print word frequencies.""" translator = str.maketrans('', '', string.punctuation) lines = load_file(filepath) counts = count_words(lines) line.translate(translator).lower() def count_words(lines): """Counts the frequency of each word in the provided lines.""" for line in lines: clean = clean_line(line) words = clean.split() for word in words: word_counts[word] = word_counts.get(word, 0) + 1 return word_counts '''</pre>	<pre>''' python # file_processor.py - Script for processing text files and counting word frequencies import string def load_file(filepath): """Reads and returns lines from the given file.""" with open(filepath, 'r') as f: return f.readlines() def clean_line(line): """Removes punctuation and converts text to lowercase.""" translator = str.maketrans('', '', string.punctuation) return line.translate(translator).lower() def count_words(lines): """Main function to process the file and print word frequencies.""" translator = str.maketrans('', '', string.punctuation) lines = load_file(filepath) counts = count_words(lines) line.translate(translator).lower() def count_words(lines): """Counts the frequency of each word in the provided lines.""" for line in lines: clean = clean_line(line) words = clean.split() for word in words: word_counts[word] = word_counts.get(word, 0) + 1 return word_counts '''</pre>	Exact Match: False
Analysis:	<p>In the Code Completion and Robustness task, both GPT-4o mini and Codestral-2501 produced functional code under Zero-Shot prompting but lacked robustness features like error handling and empty-word checks. Few-Shot prompting significantly improved code quality: both models added docstrings, handled edge cases more carefully, and Codestral introduced exception handling. Overall, Few-Shot examples guided the models to write more robust, clearer, and safer code than Zero-Shot prompts, with GPT-4o being slightly more concise and Codestral showing greater gains with examples.</p>		