Gen AI for Engineers

Prompt Engineering for Code Generation, API Design and Implementation, SQL, and Debugging

This document contains a series of prompts and examples for using Gen AI to assist with various software engineering tasks.

Date: December 15, 2024

INDEX

Prompt - Code Generation

Prompt to write a new code for new features and functionality

Prompt:

```
Act as a [Technology Name] developer. [Write a detailed description]
```

Example 1:
Act as a Python developer. Write code to read and print duplicate records from the provided CSV file.

Example 2:
Act as a JavaScript Developer, Write a program that checks the information on a form. Name and email are required, but address and age are not.

Example 3:
Act as a JavaScript Developer, Write a program that checks if a string contains a substring.

Prompt with technology stack and other details

Prompt:

```
Act as: [Enter your profile]
Technology stack: [Enter your technology stack]
Functionality: [Enter functionality]
Mandatory Fields: [Enter Fields]
Optional fields: [Enter Fields]
Task: [Write a task description]
```

Example 1:
Act as: Node.js Developer
Technology stack: Node.js, Express.js, MongoDB, Mongoose
Functionality: Newsletter
Mandatory Fields: Email
Optional fields: name
Task: Make an API that takes a name and an email address as inputs and sends back a success flag.

Example 2:
Act as: PHP Developer
Technology stack: Laravel 8, MySQL

Functionality: CAGR Calculation
Mandatory Fields: Amount and Years
Optional fields: NA
Task: Write a service that calculates CAGR.

Example 3:
Act as: PHP Developer
Technology stack: Laravel 8, MySQL
Functionality: Cron
Task: Write a cron that sends portfolio returns every day to users.

API - Design, Arch & Implementation

Prompt for API Design and Architecture:

Prompt:

```
Your task is to design and outline the architecture for a [api_type] API (e.g., REST, GraphQL, gRPC) that provides the following functionality:
[api_functionality]
The API design should include:
1. Resource modeling and endpoint structure
2. Request and response data formats (e.g., JSON, XML, Protobuf)
3. Authentication and authorization mechanisms
4. Error handling and response codes
5. Versioning and backward compatibility considerations
6. Documentation and examples
Consider the following constraints and requirements:
[requirements]
The output should be a detailed API design document, including endpoints, data models, security mechanisms, and any other relevant architectural components, along with a clear rationale for the design choices made.
```

Example:

Your task is to design and outline the architecture for a RESTful API that provides the following functionality:
- Manage user accounts (create, read, update, delete)
- Manage blog posts (create, read, update, delete)
- Manage comments on blog posts (create, read, update, delete)

The API design should include:
1. Resource modeling and endpoint structure
2. Request and response data formats (JSON)
3. Authentication and authorization mechanisms (JWT-based authentication)
4. Error handling and response codes
5. Versioning and backward compatibility considerations
6. Documentation and examples

Consider the following constraints and requirements:
- The API should follow RESTful principles and best practices
- Authentication should be required for creating, updating, or deleting resources

- Support pagination and filtering for retrieving blog posts and comments
- Maintain separation of concerns between user management, blog posts, and comments
The output should be a detailed API design document, including endpoints, data models, security mechanisms, and any other relevant architectural components, along with a clear rationale for the design choices made.

Prompt for API Implementation:

Prompt

```
Your task is to implement a [api_type] API for a [application_type] application using [programming_language] and [framework] (e.g., Node.js with Express, Python with Flask, Java with Spring Boot).
The API implementation should include:
1. Routing and endpoint handlers
2. Request and response data parsing and serialization
3. Database integration and data access layer
4. Authentication and authorization middleware
5. Error handling and logging
6. Documentation and testing
Consider the following requirements and constraints:
[requirements]
The output should be a set of properly structured and documented code files (e.g., routes, controllers, models, utilities) that implement the specified API functionality, following best practices for the chosen programming language and framework.
```

Example:
Your task is to implement a RESTful API for a blogging application using Node.js and Express.
The API implementation should include:
1. Routing and endpoint handlers for:
- User management (create, read, update, delete)
- Blog post management (create, read, update, delete)
- Comment management (create, read, update, delete)
2. Request and response data parsing and serialization (JSON)
3. Database integration and data access layer (e.g., MongoDB or PostgreSQL)
4. Authentication and authorization middleware (JWT-based)
5. Error handling and logging
6. Documentation and testing

Consider the following requirements and constraints:
- Use MongoDB as the database for storing user, blog post, and comment data
- Implement JSON Web Tokens (JWT) for authentication and authorization
- Follow best practices for Express routing and middleware
- Implement input validation and sanitization
- Include unit tests and integration tests for the API endpoints

The output should be a set of properly structured and documented code files (e.g., routes, controllers, models, utilities) that implement the specified API functionality, following best practices for Node.js and Express.

SQL

Prompt to Write SQL Queries

Prompt:

```
Your task is to generate SQL queries for the following operations on a ${database_schema} database:
1. Select all columns from the ${table_name} table.
2. Insert a new record into the ${table_name} table with the following values: ${values}.
3. Update the ${column_name} column for records where ${condition}.
4. Delete records from the ${table_name} table where ${condition}.
5. Join the ${table1_name} and ${table2_name} tables on the ${join_condition}.
The output should be a set of properly formatted SQL queries that can be executed against the specified database schema to perform the requested operations.
```

Example:
Your task is to generate SQL queries for the following operations on a blog database:
1. Select all columns from the posts table.
2. Insert a new record into the comments table with the following values: 'This is a great post!', 1, 3.
3. Update the title column for records in the posts table where post_id = 5.
4. Delete records from the comments table where comment_id > 10.
5. Join the posts and comments tables on the post_id column to retrieve all comments for each post.
The output should be a set of properly formatted SQL queries that can be executed against the specified database schema to perform the requested operations.

Prompt to Write Complex SQL Queries

Prompt:

```
Your task is to generate SQL queries for the following operations on a [database_schema] database:
1. Retrieve the [column_names] from the [table_name] table, grouped by [group_by_column] and filtered by [filter_condition].
```

```
2. Calculate the ${aggregate_function} of [column_name] for each
[group_by_column] in the [table_name] table.
3. Perform a self-join on the [table_name] table to find
[self_join_condition].
4. Create a view [view_name] that [view_definition].
5. Write a subquery to [subquery_description].
Consider the following constraints and requirements:
[requirements]
The output should be a set of properly formatted SQL queries that can
be executed against the specified database schema to perform the
requested operations, including any necessary subqueries, joins, or
aggregations.
```

Example:

Your task is to generate SQL queries for the following operations on a blog database:
1. Retrieve the title, content, and author_name from the posts table, grouped by category_id and filtered by published_date > '2022-01-01'.
2. Calculate the COUNT of comments for each post_id in the comments table.
3. Perform a self-join on the posts table to find posts with the same category_id.
4. Create a view top_posts that selects the post_id, title, and comment_count for posts with more than 10 comments.
5. Write a subquery to retrieve all posts where the author_id matches any author from the 'Technology' category.
Consider the following constraints and requirements:
- The database schema includes tables for posts, comments, categories, and authors
- Ensure proper handling of NULL values and edge cases
- Optimize queries for performance where possible
The output should be a set of properly formatted SQL queries that can be executed against the specified database schema to perform the requested operations, including any necessary subqueries, joins, or aggregations.

Fix Errors & Bugs, Generate Dummy Data

Prompt for fixing errors and bugs

Prompt:

```
Tell me how to debug the code to solve the given error.
Project: [Project name/description]
Technology Stack: [Technology Stack]
Error: [Explain the error]
```

Example:
Tell me how to debug the code to solve the given error.
Project: eCommerce
Technology Stack: JavaScript, Node.js, Express.js Stripe, MongoDB
Error: Orders get placed twice for Indian users.

Prompt:

```
I am getting the error: [Insert your error message here] Tell me how to fix it.
```

Example: I am getting the error: Cannot get strings. key_one because property key_one is missing in undefined [1]. [1] strings?: [string_key: string]: string. Tell me how to fix this

Prompt:

```
I am working on the [Enter functionality], but my code is giving the wrong answer. Tell me what the error is. Here is my code. [Paste your code here]
```

Example:
I am working on the CAGR calculation functionality, but my code is giving the wrong answer. Tell me what the error is. Here is my code.

```
function calculateCAGR(startValue, endValue, years)
{
  const cagr = (Math.pow(endValue / startValue) - 1) * 100;
  return cagr.toFixed(2);
}
```

Prompt for Generating Dummy Data

Prompt:
For training purposes only

```
I am building software and need to generate dummy data for my
functionality.
Functionality: [Explain your functionality]
Generate data: [Enter columns or data you needed]
Data format: [Enter the data format]
Number of records: [Enter number of records]
```

Example:
I am building software and need to generate dummy data for my functionality.
Functionality: Payment gateway integration
Generate data: user email, address, pin code
Data format: MySQL Queries
Number of records: 20