Documentation:

Prerequisites and Setup:

1. Stable internet connection, Google and Github account with at least one repository which you want to analyze.
2. Generate github authentication token from account settings to access your repositories and update them if required.
3. Generate Gemini API key from https://makersuite.google.com/app/apikey.
4. Users should also give their github username, repository name and filename for automated codebase analysis and improvement suggestions.

Methods:

1. *code_explainer()*:
   Takes 'code' and 'depth' parameters as input. This method explains the code in a particular depth asked by the user. 'code' can be in any programming language and 'depth' can be set to 'brief', 'deep' or any user defined depth.

2. *structure_checker()*:
   Indent the code correctly, to make sure it doesn't throw any compilation error in languages like python, also increases readability. Find mistakes in syntax of code and correct it, which include declaring the used variables which were undeclared, missing semicolons (';') or even spelling mistakes which results in compilation error, like calling 'fibonacii(n)' instead of calling 'fibonacci(n)'.

3. *time_space_complexity_analyzer()*:
   Analyze and tell time and space complexity of the code. Return the answer in two points, where the first point contains the exact complexities whereas the second point is a brief explanation about the calculations of complexities.

4. *optimization_advisor()*:
   Uses various optimization techniques for reducing space and time complexity. The optimized output code is generated by LLM for helping the user.

5. *principles_analyzer()*:
   Will take care of Software Engineering principles like KISS, DRY, YAGNI, SOLID including someOOPs concepts. Ex: this method will remove a repetitive code and make a

separate function/method for it. If some critical fields are declared in public (like password), they will be declared as private in modified code.

6. *bug_analyzer()*:
   This method will identify the possible bugs, like runtime or logic errors including infinite loops, etc. Runtime error will include division by zero, signed integer overflow, accessing illegal elements, etc.

7. *testCase_generator(testcase)*:
   Will take some sample input test cases and will suggest some stronger test cases including edge cases. For better results it is expected to give input and output in a particular format. This method will test the number of test cases and their expected output.

   Format:
   “
   Input: …..
   Output: ….
   ”