# **main\_app.py ka Poora Documentation**

## **Parichay (Introduction)**

main\_app.py file aapke "Code-Mitra" application ka mukhya hissa ya "dimaag" (controller) hai. Iska kaam hai:

1. User Interface (gui.py se) ko shuru karna aur manage karna.
2. File system mein ho rahe badlaavon par nazar rakhna (watchdog se).
3. Code ko analyze karne ke liye analyzer.py aur gemini\_client.py se baat karna.
4. User dwara diye gaye sabhi inputs (jaise button clicks) ko handle karna.
5. Sabhi kaamon ko alag-alag threads mein chalana taaki application kabhi bhi "freeze" ya "hang" na ho.

## **Class MainApplication**

Yeh is application ki sabse zaroori class hai.

### **\_\_init\_\_(self)**

* **Kaam:** Jab bhi application shuru hota hai, yeh function sabse pehle chalta hai.
* **Kya Karta Hai:**
  + self.gui = AppGUI(self): Yeh gui.py se AppGUI class ko shuru karta hai aur use batata hai ki uska controller yeh (self) class hai. Isse GUI aur logic aapas mein jud jaate hain.
  + self.observer = None: File monitor ko shuru mein khaali rakhta hai.
  + self.is\_analyzing = False aur self.analysis\_lock: Yeh ek "lock" system banata hai. Iska fayda yeh hai ki jab ek file analyze ho rahi ho, toh doosri file ka analysis shuru nahi hota. Isse "Too Many Requests" wala error nahi aata.
  + self.gui.protocol(...): Yeh sunishchit karta hai ki jab aap window ko band karein, toh on\_closing function chale.

### **run(self)**

* **Kaam:** Application ko screen par dikhana aur chalana.
* **Kya Karta Hai:** Yeh tkinter ka mukhya function mainloop() chalaata hai, jisse GUI window dikhti hai aur user ke actions ka intezar karti hai.

### **File Handling Functions**

#### **select\_folder(self)**

* **Kab Chalta Hai:** Jab aap "Select Project Folder" button dabaate hain.
* **Kya Karta Hai:**
  1. Ek folder chunne wala dialog box kholta hai.
  2. Agar aap koi folder chunte hain, toh watchdog (file monitor) ko us folder par nazar rakhne ke liye shuru kar deta hai.

#### **load\_and\_analyze\_file(self)**

* **Kab Chalta Hai:** Jab aap "Load & Analyze File" button dabaate hain.
* **Kya Karta Hai:**
  1. Ek file chunne wala dialog box kholta hai.
  2. Chuni hui file ka path (self.current\_file\_path) save kar leta hai taaki "Save" button kaam kar sake.
  3. "Save" button ko enable kar deta hai.
  4. handle\_file\_analysis function ko ek naye thread mein shuru kar deta hai.

### **Analysis Functions**

#### **handle\_live\_code\_analysis(self, content)**

* **Kab Chalta Hai:** Jab aap "Live Code Editor" tab mein type karna band karte hain.
* **Kya Karta Hai:** \_process\_analysis function ko ek naye thread mein shuru karta hai taaki live code analyze ho sake.

#### **handle\_file\_analysis(self, file\_path)**

* **Kab Chalta Hai:** Jab aap koi file load karte hain ya auto-save se koi file badalti hai.
* **Kya Karta Hai:**
  1. File ka content padhta hai.
  2. Us content ko "Live Code Editor" tab mein daal deta hai.
  3. \_process\_analysis function ko ek naye thread mein shuru karta hai.

#### **\_process\_analysis(self, content, file\_type)**

* **Kaam:** Yeh ek central function hai jo har tarah ke analysis ko handle karta hai.
* **Kya Karta Hai:**
  1. Check karta hai ki file khaali toh nahi hai.
  2. File ke type (.py ya .md) ke anusaar \_analyze\_python\_file ya \_analyze\_markdown\_file ko bulata hai.
  3. Analysis poora hone ke baad, lock ko release kar deta hai.

#### **\_analyze\_python\_file(self, content)**

* **Kaam:** Python code ko poori tarah se analyze karna.
* **Kya Karta Hai:**
  1. **AI Explanation:** Gemini API ko ek khaas prompt bhejkar code ka explanation, uske alag-alag tareeke, aur behtar banane ke sujhav maangta hai. Jawaab ko "AI Code Explanation" tab mein daal deta hai.
  2. **Error & Solution:** pylint se code ke errors nikalta hai.
     + Agar koi error nahi hai, toh "No errors found" ka message dikhata hai.
     + Agar errors hain, toh Gemini API ko ek aur prompt bhejkar un errors ka matlab aur unhein theek karne ka solution maangta hai. Jawaab ko "Errors & AI Solution" tab mein daal deta hai.

#### **\_analyze\_markdown\_file(self, content)**

* **Kaam:** Markdown file mein likhe gaye task ko samjhana.
* **Kya Karta Hai:** Gemini API ko prompt bhejkar us task ko poora karne ke steps maangta hai aur jawaab ko "Markdown Task Breakdown" tab mein daal deta hai.

### **Q&A Functions**

#### **handle\_ask\_question(self)**

* **Kab Chalta Hai:** Jab aap "Ask AI" button dabaate hain.
* **Kya Karta Hai:**
  1. Input box se aapka sawaal leta hai.
  2. "Live Code Editor" se uss samay ka code (context) leta hai.
  3. \_process\_question function ko ek naye thread mein shuru kar deta hai.

#### **\_process\_question(self, question, live\_code)**

* **Kaam:** AI se sawaal poochna.
* **Kya Karta Hai:**
  1. **Code Generation Mode:** Agar Live Editor khaali hai, toh AI ko aapke sawaal ke anusaar naya code generate karne ke liye kehta hai. Jawaab ko "Ask AI Results" tab mein dikhata hai aur saath hi Live Editor mein bhi daal deta hai.
  2. **Contextual Q&A Mode:** Agar Live Editor mein code hai, toh us code ko context banakar AI se aapka sawaal poochta hai. Jawaab ko "Ask AI Results" tab mein dikha deta hai.

### **Live Editor Functions**

#### **run\_live\_code(self)**

* **Kab Chalta Hai:** Jab aap "Run Code" button dabaate hain.
* **Kya Karta Hai:**
  1. \_execute\_code ko ek thread mein chalaakar code ka output dikhata hai.
  2. handle\_live\_code\_analysis ko doosre thread mein chalaakar code ko analyze bhi karta hai.

#### **save\_live\_code(self) aur overwrite\_file(self)**

* **Kab Chalte Hain:** Jab aap "Save As..." ya "Save" button dabaate hain.
* **Kya Karte Hain:** Live Editor ke code ko ek file mein save karte hain.

### **on\_closing(self)**

* **Kab Chalta Hai:** Jab aap application ki window ko band karte hain.
* **Kya Karta Hai:** File monitor (observer) ko surakshit roop se band karta hai taaki background mein koi process chalta na reh jaaye.

## **Class FileChangeHandler**

* **Kaam:** Yeh class watchdog library ke liye ek event handler hai.
* **on\_modified(self, event):**
  + Jab bhi monitor kiye jaa rahe folder mein koi file badalti (save hoti) hai, toh yeh function chalta hai.
  + Yeh handle\_file\_analysis function ko bulakar us file ka analysis shuru kar deta hai.
  + Ismein ek chota sa cooldown (1.5 seconds) hai taaki ek hi save par baar-baar analysis na ho.

# gui.py ka Poora Documentation

## Parichay (Introduction)

Yeh file application ke poore user interface (GUI) ko banane aur manage karne ke liye zimmedar hai. Ismein Tkinter library aur uske behtar ttk themed widgets ka istemaal kiya gaya hai taaki application ko ek modern, dark theme look mil sake.

## Class AppGUI

Yeh is file ki mukhya class hai, jo Tkinter ki Tk class se inherit karti hai.

### \_\_init\_\_(self, controller)

* **Kaam:** Application ki mukhya window ko banata hai.
* **Kya Karta Hai:**
  + self.controller = controller: Yeh main\_app.py se mile controller ka reference save kar leta hai, taaki GUI ke buttons main\_app.py ke functions ko bula sakein.
  + Window ka title aur size set karta hai.
  + \_configure\_styles() ko bulakar UI ko sundar banata hai.
  + \_create\_widgets() ko bulakar sabhi buttons, tabs, aur text boxes ko screen par banata hai.

### \_configure\_styles(self)

* **Kaam:** Application ko dark theme look dena.
* **Kya Karta Hai:**
  + ttk widgets (jaise Button, Tab, Label) ke liye custom styles banata hai.
  + Ismein har cheez ka color, padding, aur font set kiya jaata hai taaki UI professional dikhe.

### \_create\_widgets(self)

* **Kaam:** Application ke sabhi visual hisson ko banana aur screen par sahi jagah par lagana.
* **Kya Karta Hai:**
  + **Top Frame:** "Select Folder" aur "Load & Analyze File" buttons ko rakhta hai.
  + **Q&A Frame:** "Ask AI" ka input box aur button banata hai.
  + **Notebook (Tabs):** Application ke mukhya hisse, yaani sabhi tabs (Live Code Editor, AI Code Explanation, etc.) ko banata hai.
  + **Live Code Editor Tab:**
    - Yeh ek khaas tab hai jismein PanedWindow ka istemaal kiya gaya hai. Isse user code editor aur output area ka size apni zaroorat ke anusaar chota-bada kar sakta hai.
    - Ismein "Run Code", "Save", aur "Save As..." buttons jode jaate hain.
  + **Status Bar:** Neeche ek patti banata hai jo user ko batati hai ki app abhi kya kar raha hai (jaise, "Analyzing...").

### UI Interaction Functions

#### on\_key\_release(self, event=None) aur trigger\_live\_analysis(self)

* **Kaam:** Live editor mein automatic analysis ka feature laagu karna.
* **Kaise Kaam Karta Hai (Debouncing):** Jab user live editor mein type karta hai, toh on\_key\_release function har key press par chalta hai. Lekin, yeh turant analysis shuru nahi karta. Yeh 1.5 seconds ka timer set karta hai. Agar user 1.5 seconds ke andar koi aur key daba deta hai, toh purana timer cancel ho jaata hai aur naya timer shuru ho jaata hai. Jab user 1.5 seconds tak kuch nahi likhta, tabhi trigger\_live\_analysis function chalta hai. Isse baar-baar API request nahi jaati.

#### load\_content\_to\_editor(self, content)

* **Kaam:** File se padhe gaye content ko "Live Code Editor" tab mein daalna.
* **Kya Karta Hai:**
  1. Editor ko pehle poori tarah se saaf karta hai.
  2. Naya content usmein daal deta hai.
  3. User ka focus apne aap "Live Code Editor" tab par le aata hai.

### Thread-Safe Update Functions

#### update\_display(self, target\_widget, content)

* **Kaam:** Background thread se aayi jaankari ko GUI mein surakshit roop se update karna.
* **Kyun Zaroori Hai:** Tkinter mein, aap seedhe ek background thread se GUI ko nahi badal sakte. Aisa karne se app crash ho sakta hai. Yeh function self.after(0, ...) ka istemaal karke update ke kaam ko main GUI thread mein bhej deta hai, jisse app surakshit rehta hai.
* **Kya Karta Hai:** Yeh main\_app.py se bataye gaye tab (jaise "explanation", "errors") mein naya content daal deta hai.

#### update\_status(self, message) aur update\_folder\_label(self, path)

* **Kaam:** Yeh bhi thread-safe functions hain jo status bar aur folder label ko update karte hain.

# analyzer.py ka Poora Documentation

## Parichay (Introduction)

Yeh file un sabhi analysis ke kaamon ko karti hai jinke liye AI ki zaroorat nahi hai. Iska kaam hai code ko sthaniya (locally) jaanchana aur file system se jaankari padhna. Isse main\_app.py saaf-suthra rehta hai.

## Functions

### run\_pylint(file\_path: str) -> str

* **Kaam:** Diye gaye Python file par pylint tool ko chalana.
* **Input:** Python file ka poora path (location).
* **Output:** pylint dwara di gayi poori report (text format mein).
* **Kaise Kaam Karta Hai:**
  1. Yeh Python ke subprocess module ka istemaal karta hai. subprocess ek naya process shuru karke usmein command chalaane ki suvidha deta hai, bilkul waise hi jaise aap apne computer ke Terminal mein command likhte hain.
  2. Yeh pylint <file\_path> command chalaata hai.
  3. capture\_output=True se yeh pylint ke saare output (jo Terminal mein dikhta hai) ko capture kar leta hai.
  4. --exit-zero ek khaas flag hai jo pylint ko batata hai ki agar use code mein galtiyan milein, toh bhi program ko error dekar band na kare.
  5. Ant mein, yeh capture ki gayi report ko text ke roop mein main\_app.py ko wapas bhej deta hai.

### read\_file\_content(file\_path: str) -> str

* **Kaam:** Kisi bhi file ke poore content ko surakshit roop se padhna.
* **Input:** File ka poora path.
* **Output:** File ka poora content (text format mein).
* **Kaise Kaam Karta Hai:**
  1. Yeh Python ke standard file handling (open(...)) ka istemaal karta hai.
  2. encoding='utf-8' ka istemaal yeh sunishchit karta hai ki file mein Hindi ya koi aur special character hone par bhi woh theek se padha jaaye.
  3. Yeh poore content ko ek hi baar mein padhkar main\_app.py ko wapas bhej deta hai.

# gemini\_client.py ka Poora Documentation

## Parichay (Introduction)

Yeh file application ka woh hissa hai jo Google Gemini API se baat karke AI ki shakti ka istemaal karta hai. Is file ko alag banane ka fayda yeh hai ki API se judi saari settings aur logic ek hi jagah par rehti hain.

## Functions

### query\_gemini(prompt: str) -> str

* **Kaam:** AI se koi bhi sawaal (prompt) poochna aur uska jawaab lana.
* **Input:** Ek prompt (text format mein sawaal).
* **Output:** AI dwara diya gaya jawaab (text format mein).
* **Kaise Kaam Karta Hai:**
  1. **API Key Check:** Sabse pehle, yeh check karta hai ki user ne apni API key file mein daali hai ya nahi.
  2. **Request Banana:** Yeh requests library ka istemaal karke Gemini API ko ek HTTP POST request bhejta hai. Is request ke body mein user dwara bheja gaya prompt hota hai.
  3. **Error Handling (Exponential Backoff):** Yeh is function ka sabse zaroori hissa hai.
     + Agar API **Error 429 (Too Many Requests)** bhejta hai, toh yeh function turant haar nahi maanta.
     + Yeh thodi der (shuru mein 1 second) rukta hai aur fir se request bhejta hai.
     + Agar fir se wahi error aata hai, toh yeh agli baar dugna samay (2 seconds, fir 4 seconds) rukta hai.
     + Aisa karne se API ko free hone ka samay mil jaata hai aur application crash nahi hota.
  4. **Jawaab Padhna:** Jab API se safaltapoorvak jawaab mil jaata hai, toh yeh us jawaab (JSON format mein) ko aache se padhkar usmein se zaroori text nikalta hai.
  5. Ant mein, yeh saaf-suthre text jawaab ko main\_app.py ko wapas bhej deta hai.

# Code-Mitra v3.0 - Aapka Personal AI Coding Assistant

<!-- Yahan aap apne app ka ek screenshot daal sakte hain -->

Code-Mitra ek 100% Python mein bana hua desktop application hai jo developers ko coding karte samay real-time mein madad karta hai. Yeh ek AI-powered coding saathi ki tarah kaam karta hai jo code ko samjhata hai, galtiyan dhoondhta hai, unka solution deta hai, aur aapke sawaalon ke jawaab bhi deta hai.

## Yeh Kya Problem Solve Karta Hai?

Yeh application developers ki un samasyaon ko hal karta hai jinke liye unhein baar-baar apne code editor se bahar jaana padta hai. Isse unka focus bana rehta hai, samay bachta hai, aur productivity badhti hai.

* **Automatic Code Analysis:** Code mein chhipi hui galtiyan aur style se judi samasyaon ko dhoondhna.
* **AI-Powered Explanation:** Complex code ke logic ko aasan bhasha mein samajhna.
* **Error Solutions:** Sirf error batana nahi, balki AI ki madad se unka sahi solution bhi dena.
* **Intelligent Q&A:** Code se jude sawaalon ke jawaab paana ya naya code generate karwana, sab ek hi jagah par.

## Mukhya Features (Key Features)

* **Live Code Editor:** Ek built-in editor jismein aap code likh sakte hain, use run karke output dekh sakte hain, aur file mein save bhi kar sakte hain.
* **Real-time Analysis:** Jaise hi aap live editor mein type karna band karte hain, app apne aap code ko analyze karna shuru kar deta hai.
* **File Monitoring:** Ek poora folder select karein aur us folder mein save hone wali har file ka automatic analysis paayein.
* **Enhanced AI Explanation:** AI aapko code ka matlab, use karne ke alag-alag tareeke, aur behtar banane ke sujhav deta hai.
* **AI Error Solver:** pylint dwara dhoondhi gayi galtiyon ko AI samjhata hai aur unhein theek karne ke liye corrected code bhi deta hai.
* **Dual-Mode "Ask AI":**
  + **Code Generation:** Agar editor khaali hai, toh AI se naya code generate karwayein (jaise, "factorial ka code do").
  + **Contextual Q&A:** Agar editor mein code hai, toh us code se juda koi bhi sawaal poochein.
* **Markdown Task Planner:** .md file mein apne project ka idea likhein aur AI se use poora karne ke steps paayein.

## Technology Stack (Istemal Ki Gayi Technologies)

Yeh poora application sirf Python aur uski standard libraries ka istemaal karke banaya gaya hai.

* **Bhasha (Language):** Python 3
* **GUI (User Interface):** Tkinter (ttk themed widgets ke saath)
* **File Monitoring:** watchdog library
* **Code Analysis:** pylint library
* **AI Integration:** Google Gemini API
* **API Communication:** requests library
* **Concurrency:** threading aur subprocess modules

## Setup aur Istemal (Setup and Usage)

Is application ko apne computer par chalane ke liye neeche diye gaye steps follow karein.

### 1. Zaroori Cheezein (Prerequisites)

* Aapke computer mein **Python 3.8** ya usse naya version install hona chahiye.
* Aapke paas ek **Google Gemini API Key** honi chahiye.

### 2. Installation

1. Is project ke code ko download karein ya git clone karein.
2. Apne Terminal ya Command Prompt mein project ke folder ke andar jaayein.

Neeche di gayi command chalaakar zaroori libraries install karein:  
pip install -r requirements.txt

### 3. API Key Configuration

1. Project mein gemini\_client.py naam ki file kholein.

File ke shuru mein di gayi neeche di gayi line mein apni asli Gemini API key daalein:  
API\_KEY = "YOUR\_GEMINI\_API\_KEY\_HERE"

1. File ko save karein.

### 4. Application ko Run Karein

Terminal mein project folder ke andar se neeche di gayi command chalaayein:

python main\_app.py

Isse "Code-Mitra" ki GUI window khul jayegi.

### 5. Istemal Kaise Karein (How to Use)

* **Live Coding:** "Live Code Editor" tab mein seedhe code likhna shuru karein. "Run Code" button se output dekhein aur "Save As..." se file save karein.
* **File Analysis:** "Load & Analyze File" button se koi bhi .py ya .md file chunein. Uska code apne aap Live Editor mein aa jayega aur analyze ho jayega.
* **Code Generation:** Live Editor ko khaali rakhein, neeche box mein likhein "fibonacci series ka code do" aur "Ask AI" dabayein.
* **Sawaal Poochna:** Live Editor mein code likha rehne dein aur neeche box mein us code se juda sawaal poochein. Jawaab aapko "Ask AI Results" tab mein milega.