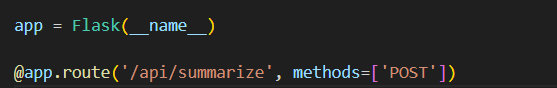
# Code explanation with screenshots

**1.1-Explanation of the Flask API for Document Summarization**

**(Backend)**

1. Flask Application Setup:

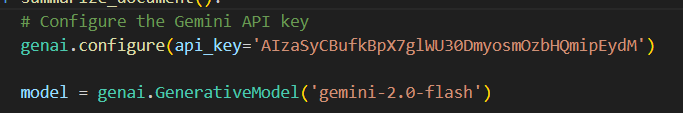
Create flask app instant , defines a single POST endpoint



/api/summarize

2. Gemini AI integration :

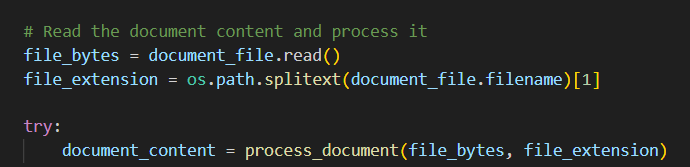
Configures the Gemini API with a provide key , use the gemini-2.0-flash model for the



fast summarization

3. Document Processing Flow :

Receives uploaded document files, process different document formats (via Process\_document) function from document.py code , then Generates summaries with customable length and detail level.



4. Define the prompt for generating the summary

Then try to generate the summary using Gemini

Model ,Execution Runs on all interfaces(0.0.0.) on

A screen shot of a computer program

AI-generated content may be incorrect.

Port 5000

A screen shot of a computer code

AI-generated content may be incorrect.

**1.2-Flutter API Client for Document Summarization**

For interacting with Flask document summarization API

The main Class (APIClient)contain parameters , summaryLength ,detailLevel,filePath,

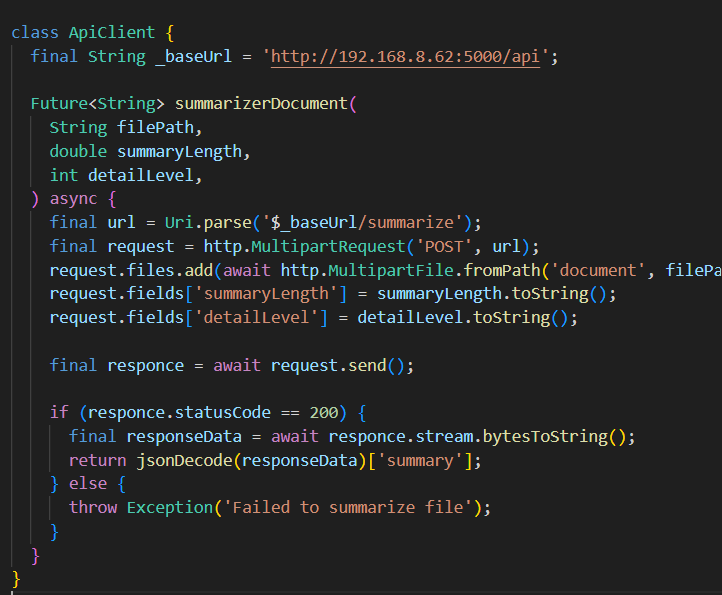
**First** creates a multipart request, by setting up the request to /api/summarize endpoint

Then attaches the document file using MultipartFile.fromPath() filnally adds summart parameters as form fields.

**Second Send the request to execute the HTTP call**

**Lasley** Process the response For success (statusCode 200) Reads response stream

, decode JSON to extract the summary field ->**figure 1.2**

figure 1.2

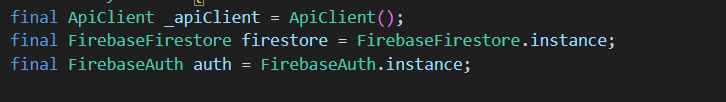
**1.3 : Flutter SummaryService Class Explanation**

This class (SummaryService ) (figure 1.3 )integrate Firebase

figure 1.3

Firestor , Firebase Auth and APiClient to handel

Document summarization and storage



These Lines for initilalization , ApiClient used

To send files to summarization Api,

FirebaseFirestore stores summaries in structures

Way , FirebaseAuth manage user authentication

To store summaries per user

And For the code moethods

-summarizeFile call function suumarizerDocument to send the file to Flask API and returns the generated summary as text (String)

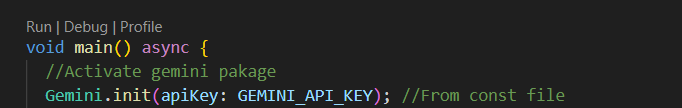
-storSummary fetchSummaries,deleteSummary all these functions deal with firebase DB

**2.Flutter Gemini Chatbot Screen**

This code implementation is a chatbot interface using Google’s Gemini AI with features like text/image messaging, editing, and clipboard copying

For Gimini, we use the package (flutter\_gemini: ^3.0.0)

Need first Initialize Gemini in the main function (figure 2.1)

figure 2.1

The core Components :

**A. State Management**: uses SattefullWidget to manage

Messages (list of chatMessage) objects, currentUser, Gemini(Chat participants' profiles), and Gemini. the instance is an AI model handler (Figure 2.2)

**B. UI Structure** used DashChat package (figure 2.3)

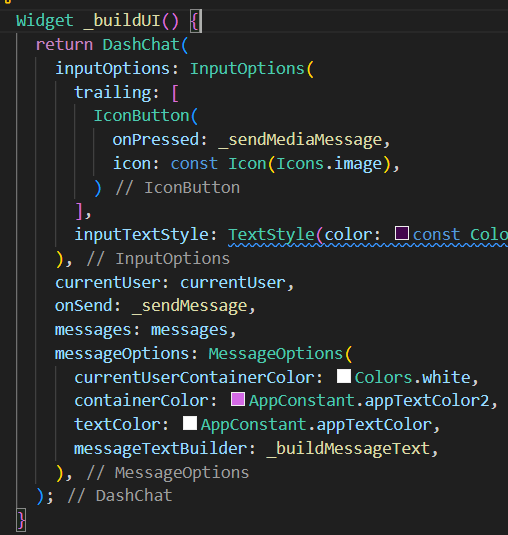
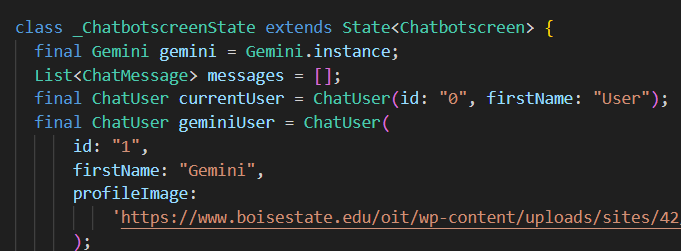
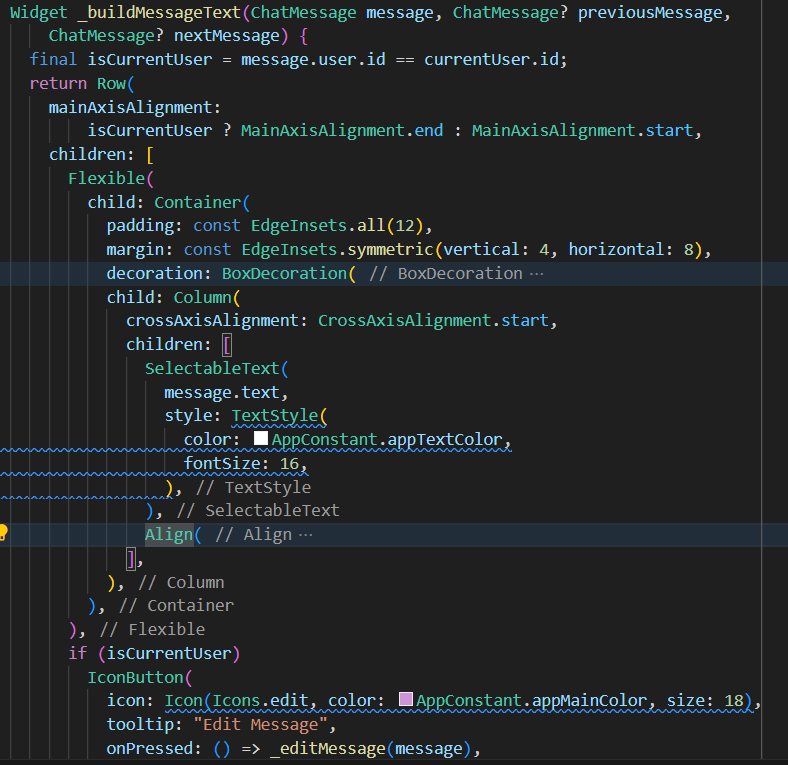
figure 2.3

Figure 2.2

figure 2.4

**C. Message** Widget(\_buildMessageText) (figure2.4)

This function is for bubble styling copy button, and Edit

Other Methods:

-\_**sendMessage**():handel text/image messages by adding user

Messge to messages and if image is attached,read it as Unit8List

And for strean gemini’s respomce updage UI in realtime and

Appends response to exusting messages

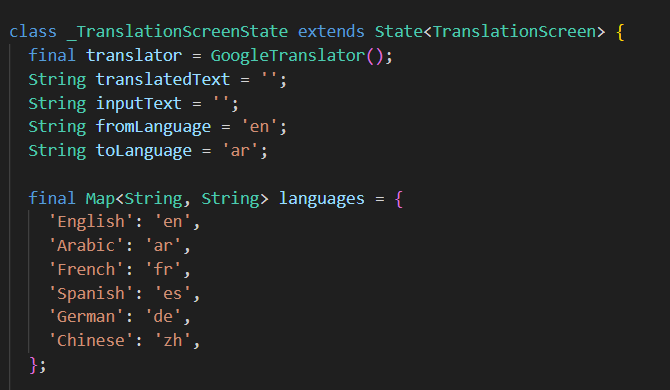
-\_**sendMediaMEssage**():this method for opening camera/gallery

Via package (ImagePicker) and sends the image to Gemini with a default prompt :”Describe this picture?”

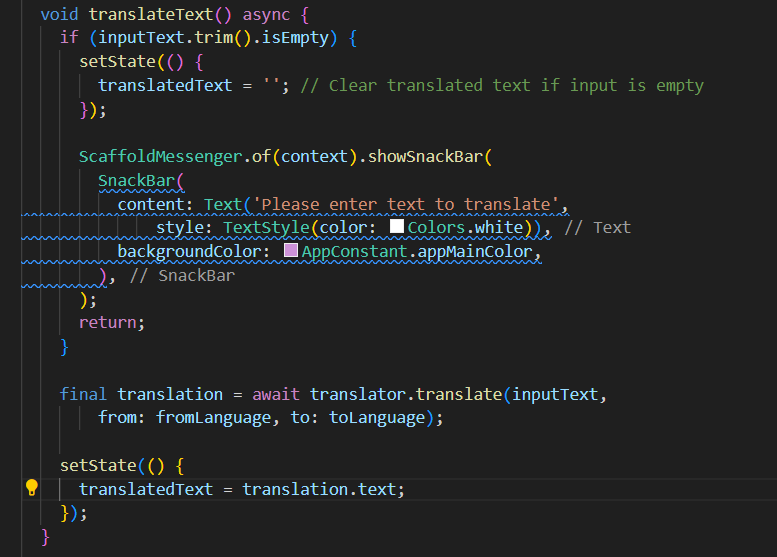
**3-Flutter Translation API**

This code implements a language translation interface using Googles Translation

API(**translator package**), the code supported 6 languages(figure 3.1)

figure 3.1

The core function is **translateText**():

Figure 3.2



This function checks for empty input ->shows snakbar if emply

Call Google Translator , updates translatedText with result

Figure(3.2)

**4-Flutter Notes screen**

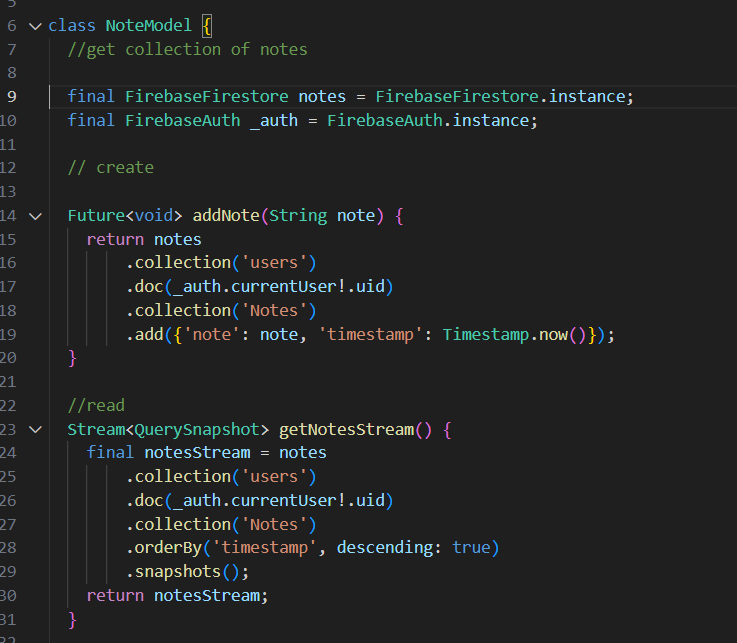
A screen shot of a computer code

AI-generated content may be incorrect.figure 4.1

**T**his class contain NoteModel and

Textcontroller , figure 4.1

For NoteModel:

figure 4.2

This class use FirebaseFirestore to save , update,

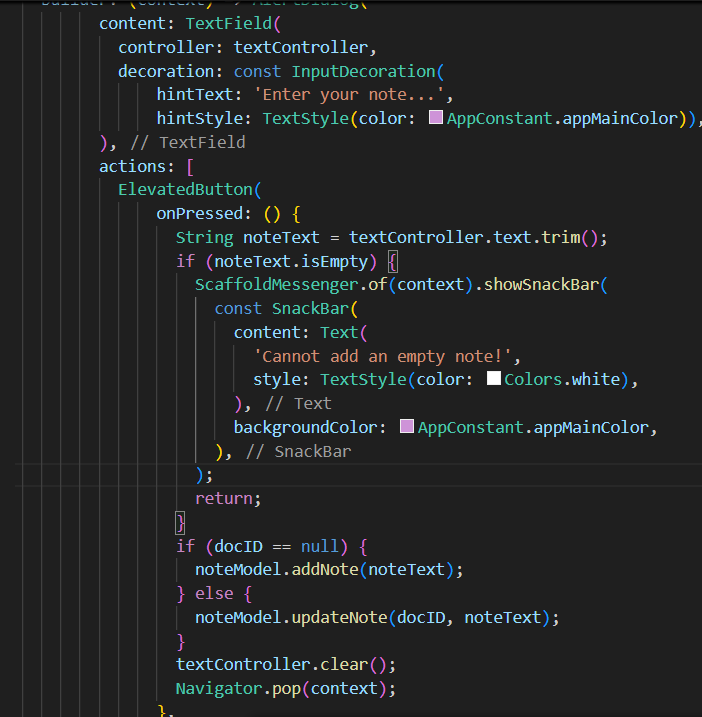
Delete notes , and firebaseAuth for each current user

This class have multiple methods as shown in figure4.2

And for \_NoteScreenState used the NoteModel

For controlling UI and how the user deals with nots

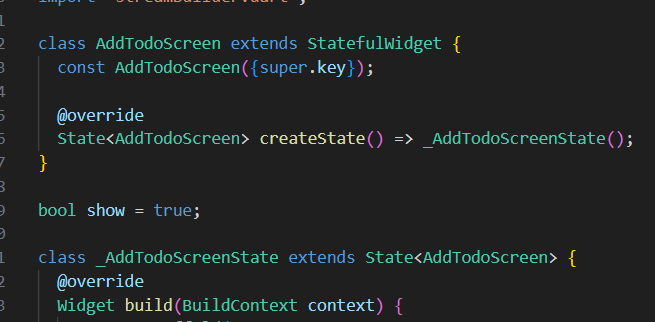
(Figure 4.3)

figure 4.3



**5-Flutter TO-DO-List screen**



Figure5.1

**The AddTodoScreen** class is the main UI class

That contains all other components to let the user

Making to-do lists,

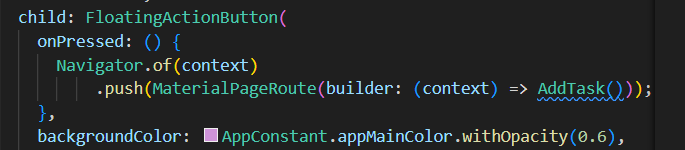
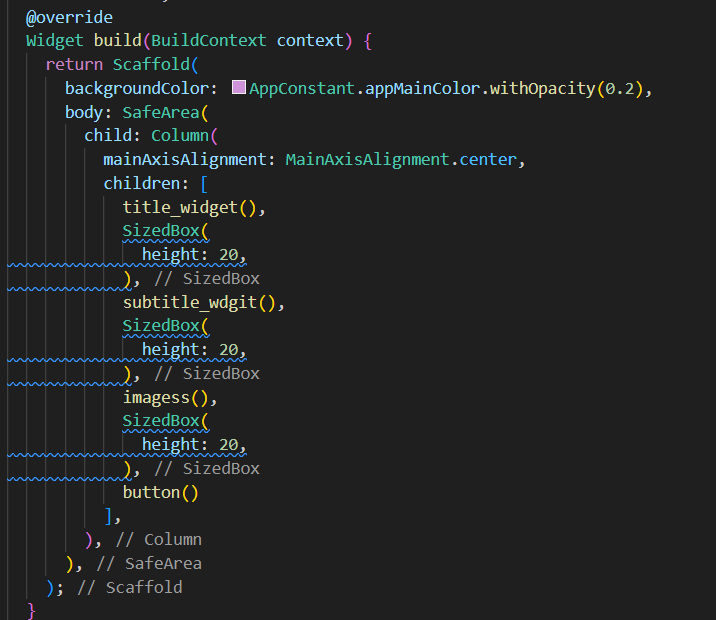
figure 5.2

Figure 5.2: Button to let the user add a Task

figure 5.3

This component is shown figure5.3

let the user enter task title ,task subtitle,images for

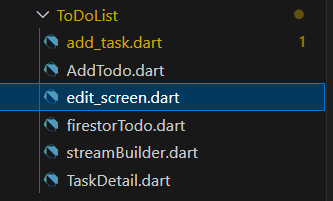
one task then upload this task to firebasefirestor

if the user click add and Cansel if the user want to

other functions for retreveing tasks , tell if the task

is done or not , display all task details and edit them when the user

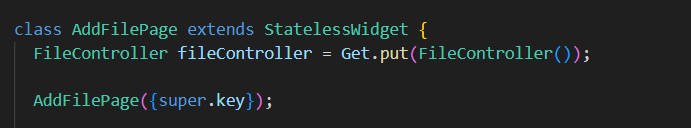
click on any of the tasks -figure 5.4-

figure 5.4

**6-Flutter Add file Screen**

This code contains multiple models and controllers to help the user

Uploading files with information

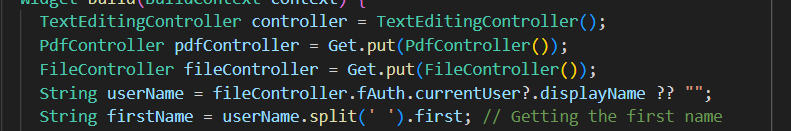
figure6.1

**Class Addfile** is the main function that contain

all the other component and controller-fig6.1

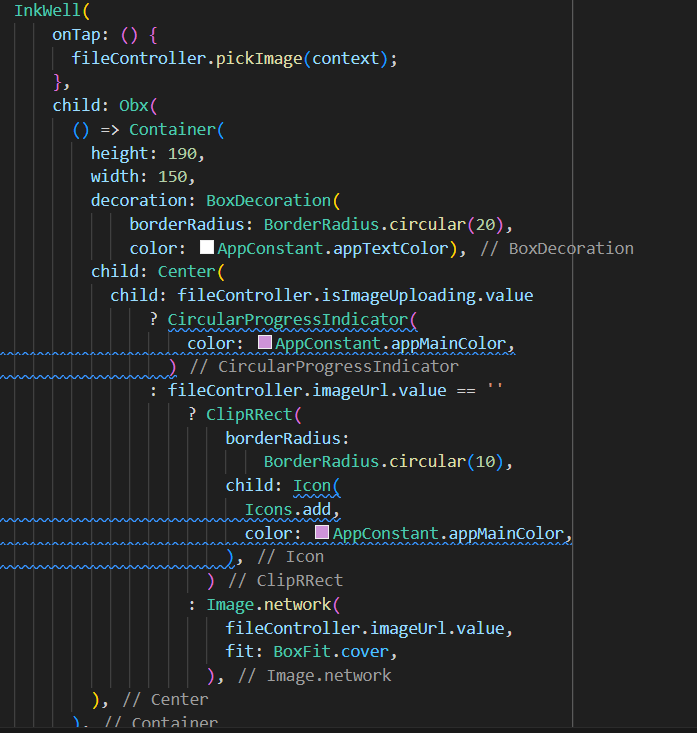
As shown in figure 6.2 multiple controllers

appears + for fetching username we use

figure 6.2

fireAuth to get the first name for the current user

1-The user **can upload cover page** for the uploaded

figure 6.3

File , but there is a default image if the user does not

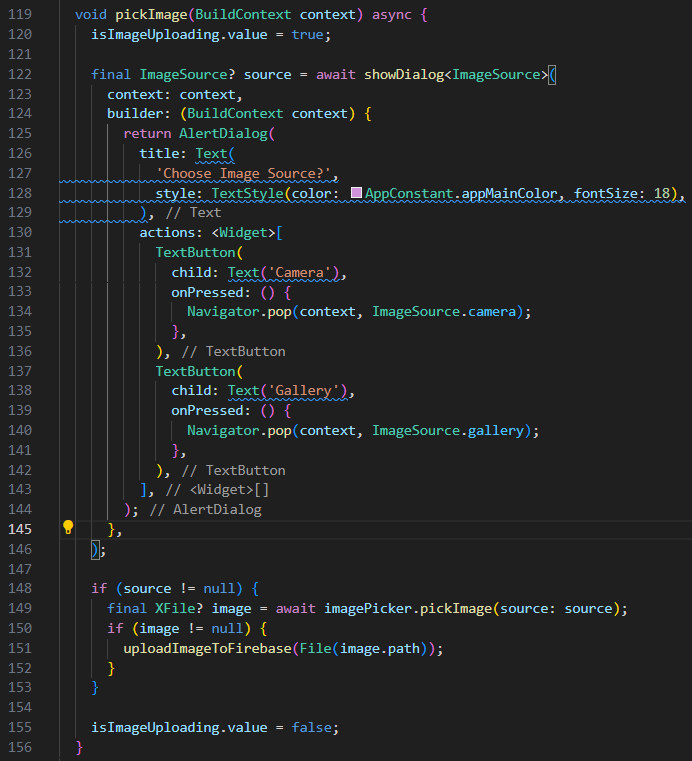
Pick one, =>figure 6.3

For **pickImage function** at the FileController that used at

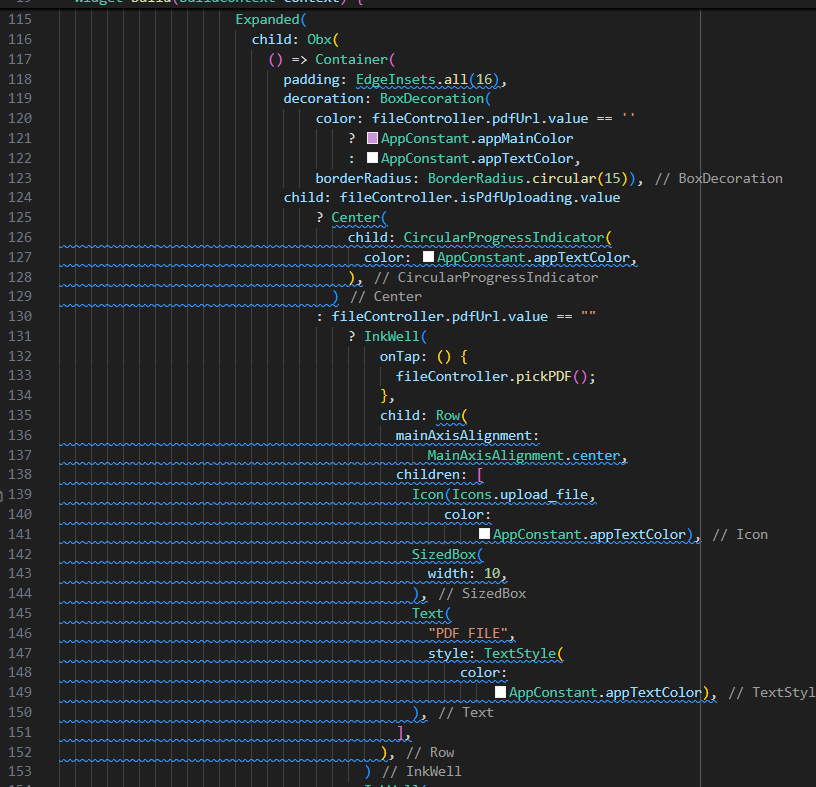
Figure 3.6 , the implementation is at figure 6.4

It asks the user from where you want to pick the image gallery or

Camera using **ImagePiker package**

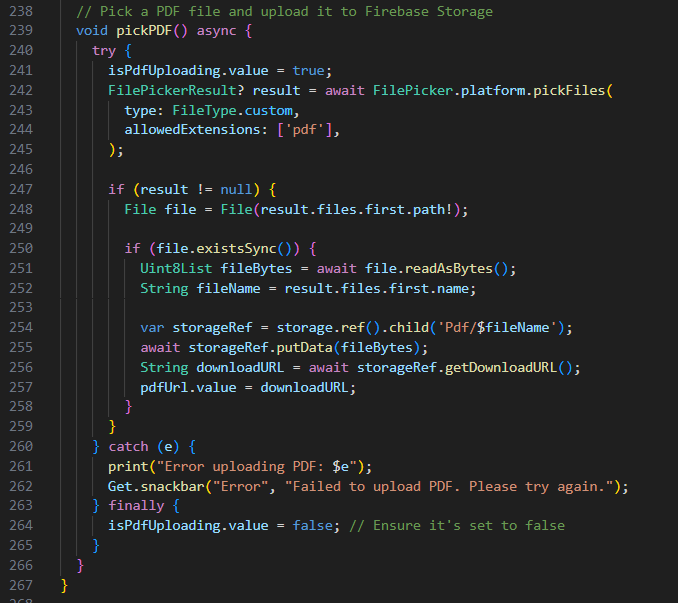
figure 6.4

2-upload pdf file figure 6.5

figure 6.5

For this part function PickPDF that in

FileController as figure 6.6

figure 6.6



It used **FilePicker package**

**That allows here only to upload pdf files**

Using **package File**

Fetch path of the file if it is exits

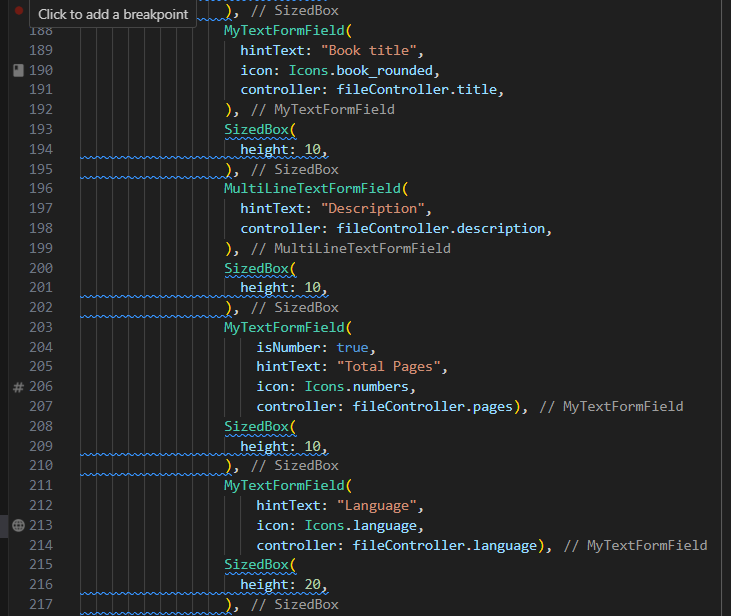
Read the file as stream of bytes

Store the file at storage with reference

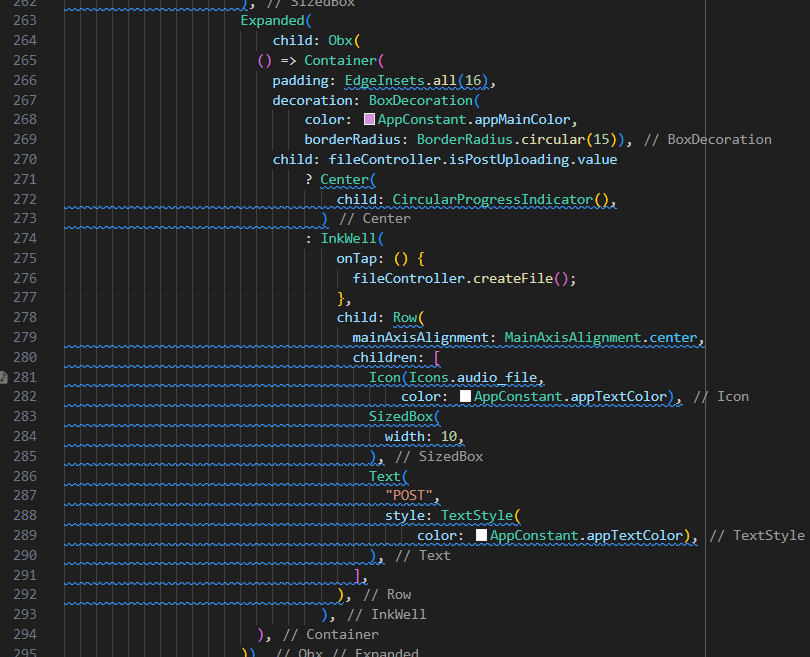
PDF at firebase Firestore

3-Filling information for the uploaded

Pdf file as figure 6.7

Figure 6.7

4- upload input information to firebase cloud

figure6.8

When the user click POST button figure 6.8, here

FileController with createFile method figure 6.9

By filling the FileModel that contains the fields

structed at the firebase cloud , then add it to

A screen shot of a computer program

AI-generated content may be incorrect.figure 6.9

File collection and userFiles collection to maintain

Space foe each user their files alone in one collection