[CO-PO Attainment Generator](https://github.com/jeryjs/co_po_attainment_generator/)

[co\_po\_generator - Dart API docs (jeryjs.github.io)](https://jeryjs.github.io/co_po_attainment_generator/)

# Overview

CO-PO Attainment Generator is a excel dependent calculator built using the Flutter framework, which allows for cross-platform development, meaning the application can be made to run on all platforms like web, windows, Linux, mac, android, etc. As of now, only windows is targeted.

The application is currently in its alpha stage, with a version of 0.5.0-alpha. This indicates that while the application is functional, it is still in the early stages of development and may contain bugs or incomplete features.

The purpose of this application is to automate the process of generating an excel sheet for calculating CO-PO attainment Automating this process can significantly reduce the manual effort required and increase the accuracy of the results.

The application uses the following dependencies to provide its functionality:

* file\_picker: This package is used for picking files from the file system. It allows the user to select files that they want to process for CO-PO attainment.
* flutter: This is the Flutter SDK, which provides the core framework for building the application.
* google\_generative\_ai: This package is used for using Google’s Gemini AI to extract *question number, mark and CO level* from the user’s selected question paper.
* image: This package is used for handling images that are to be passed to the Gemini AI for question paper extraction.
* open\_file: This package is used for opening files, used for viewing the generated excel files.
* path: This package is used for handling file paths, which is essential for file operations like reading and writing data.
* path\_provider: This package is used for finding commonly used locations on the filesystem, which can be useful for storing temporary files or user preferences.
* shared\_preferences: This package is used for persisting data, such as user settings or recent files.
* shimmer: This package is used for adding a shimmer effect animation, which can be used to indicate loading or processing status in the UI.
* msix: This package is used for generating a msix installer for the application.

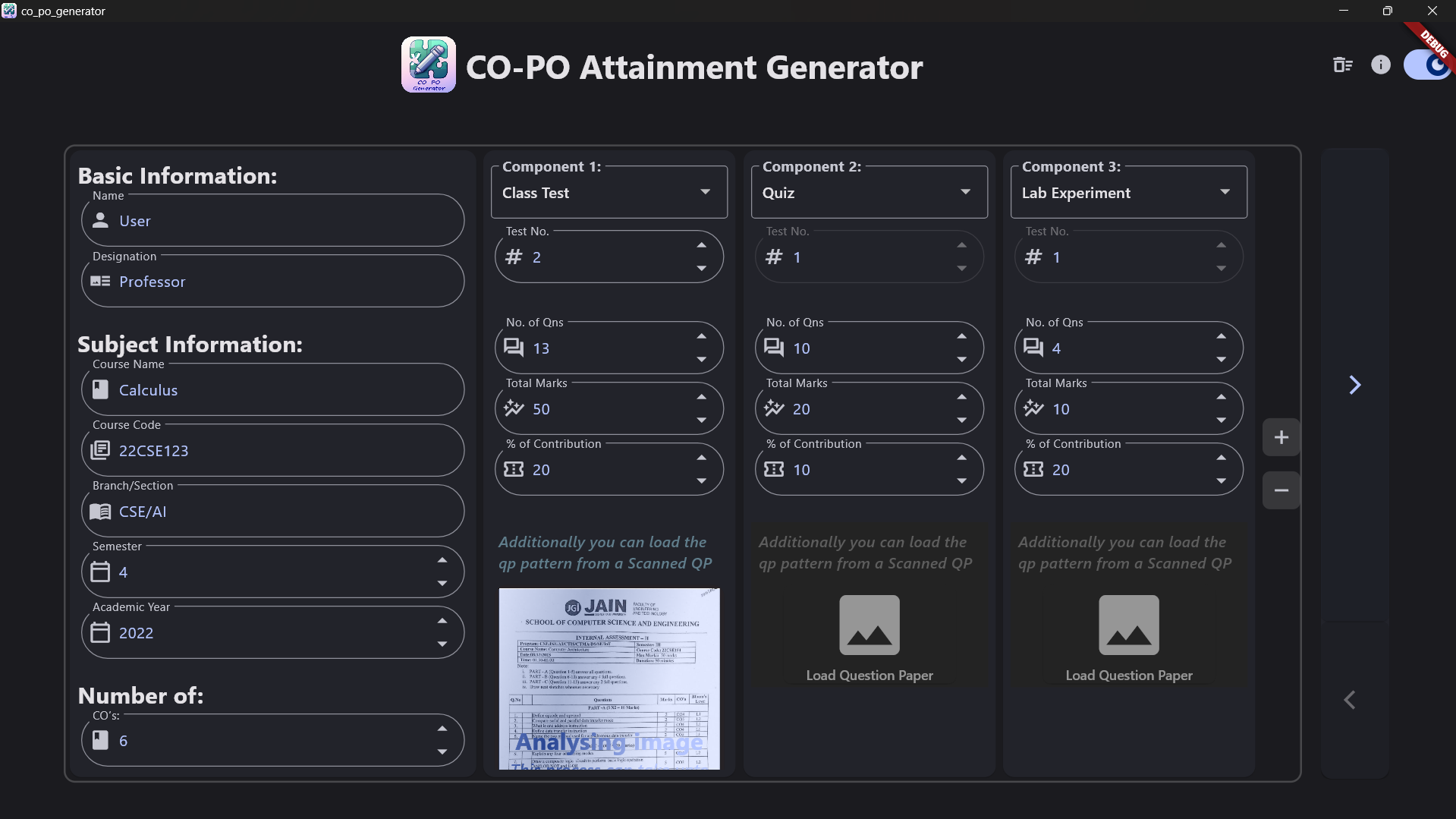
The main application's code is organized into the lib directory, which contains the main Dart code for the application, and an assets directory, which contains various resources like images, sample data, and API related assets.

Inno setup 6 and the msix package are used for generating installers for this app. Msix is windows’ latest installer packaging method.

# Working

## Pages

### DetailsPage:



The first screen in [lib/screens/1\_DetailsPage](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is the **DetailsPage**. This page is used to gather various details from the user. It contains a list of components and a widget for getting details.

The DetailsPage class has a ‘*components list'* that holds instances of GetComponent. Each GetComponent represents a component on the start page and stores information regarding the component and has a QpAnalyser.

The **QpAnalyser** uses Google’s Gemini AI to extract the user’s selected question paper’s details and store it in the *Json* format as shown In the example below-

{

    "compartment": "Internal Assessment",

    "test\_no": "2",

    "course\_name": "Computer Architecture",

    "course\_code": "22CSE154",

    "semester": "3",

    "qp\_pattern": [

        { "Q": 1, "C": 4, "M": 2 },

        { "Q": 2, "C": 4, "M": 2 },

        { "Q": 3, "C": 4, "M": 2 },

        { "Q": 8, "C": 5, "M": 5 },

        { "Q": 10, "C": 4, "M": 5 }

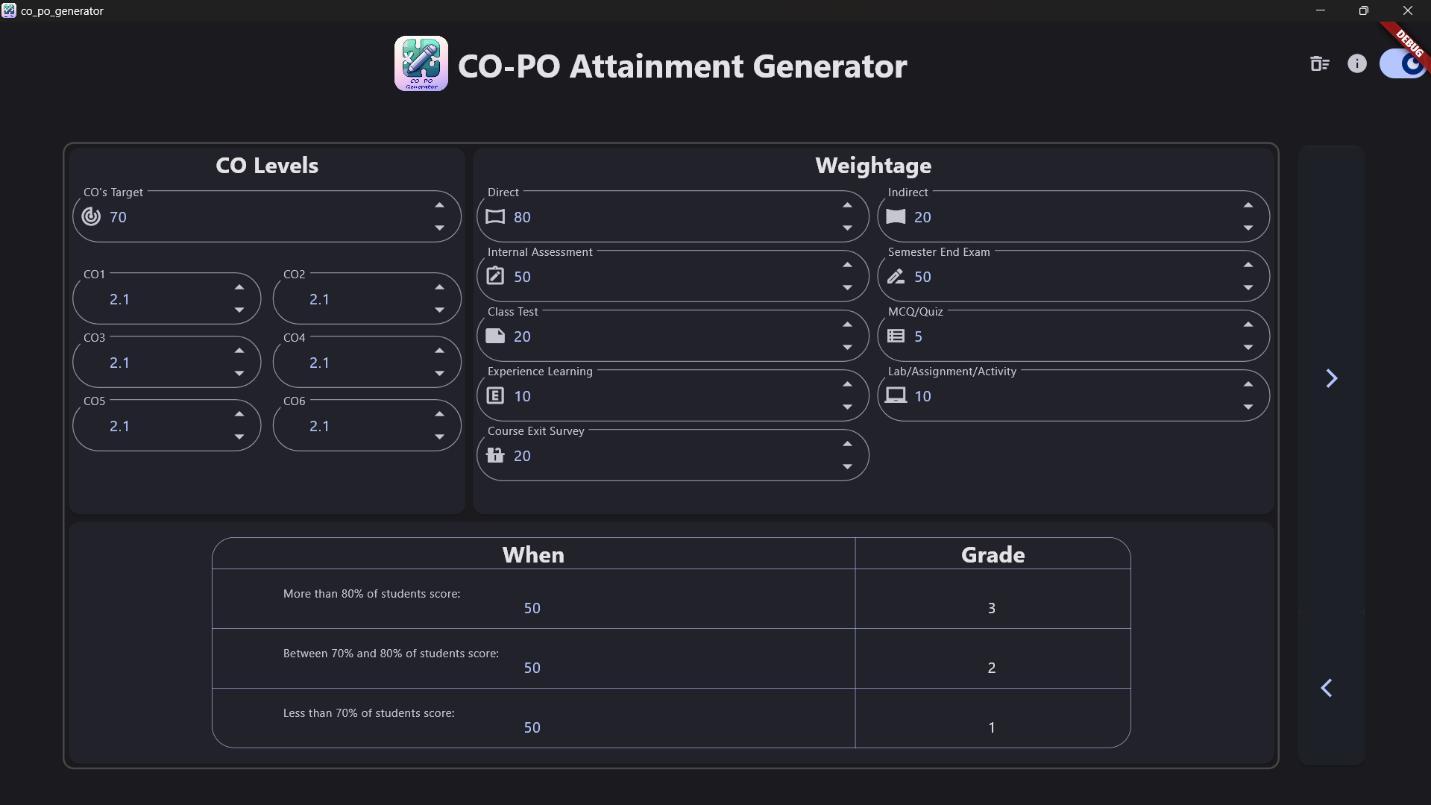
    ]

}

The DetailsPage also contains a getDetailsWidget which allows the user to input details such as name, designation, course information, etc. The isFilled method in GetDetails is used to check if the form is filled and validates all input fields.

The DetailsPage uses these components and the getDetailsWidget to gather necessary details from the user. The isFilled method in DetailsPage checks if all the components and the getDetailsWidget are filled.

### WeightagePage:



The second screen in [lib/screens/2\_WeightagePage](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is the **WeightagePage**. In this page the user can input weightages for different parameters. It contains three forms, each with its own key for validation.

The state of the forms is managed by the \_WeightagePageState class.

The \_WeightagePageState class manages the state of the text controllers and the cell mapping. It has a method \_restoreControllers which restores the text controllers from shared preferences. This allows the user to leave the page and come back without losing their input.

The WeightagePage widget uses a list of form keys for validation. It has a method isFilled which checks if all input fields are filled and validates them. It returns true if all fields are filled and valid, false otherwise.

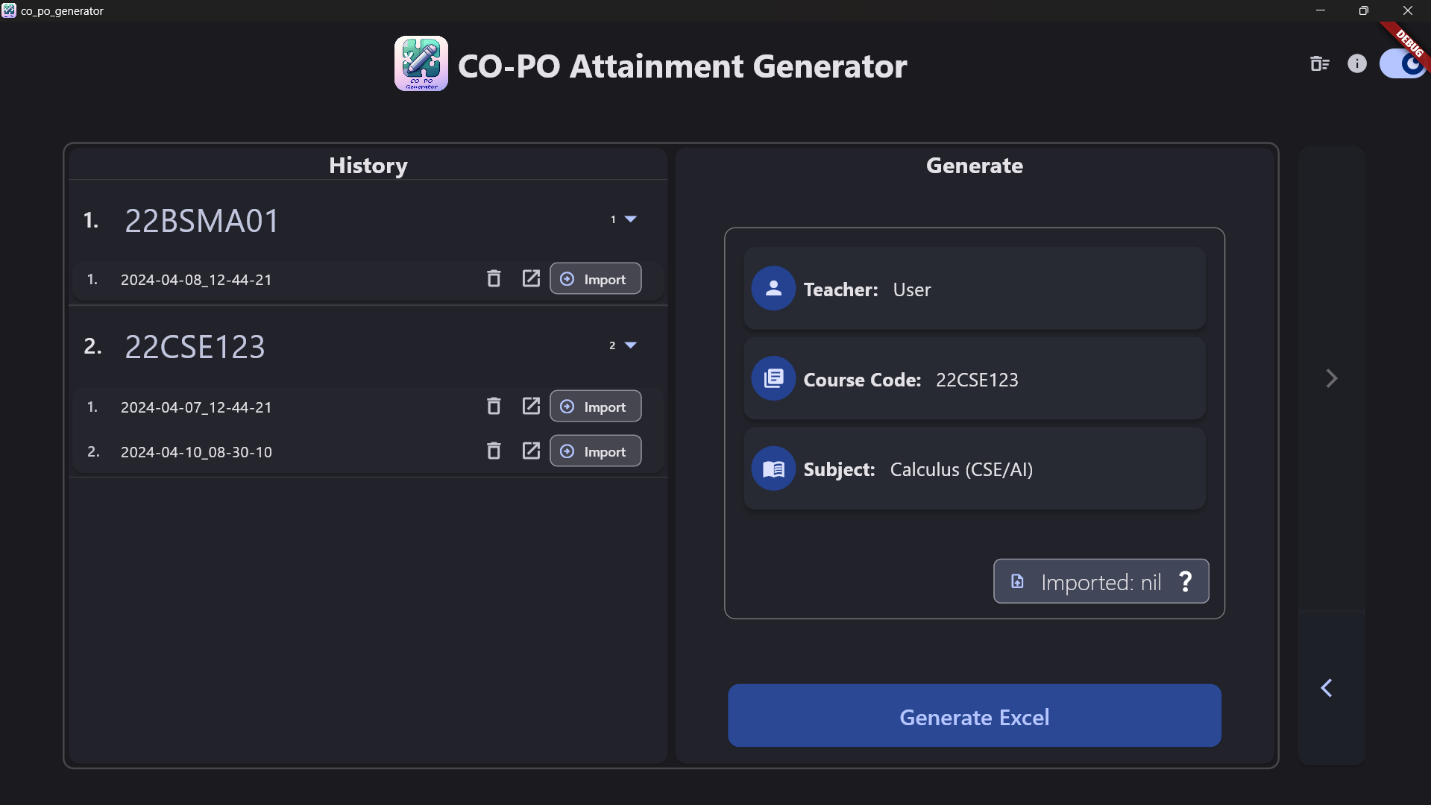
The WeightagePage uses the CellMapping class from [lib/models/cell\_mapping.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) to manage the cell mapping for the weightage. The CellMapping class has methods to get and set the weightage mapping with the provided controllers.

The MTextForm widget from [lib/components/m\_text\_form.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is used for the input fields. The MTextForm widget is a custom text form field widget.

The NextButton widget from [lib/components/widgets.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is used for the next button which is a custom button widget.

The WeightagePage uses the SharedPreferences class from the shared\_preferences package to store and retrieve the state of the text controllers. The SharedPreferences class provides a persistent store for simple data.

### GeneratePage:



The third screen in [lib/screens/3\_GeneratePage](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is the **GeneratePage**. This is the page where the Excel file generation takes place. It contains two main sections: a history view and a generate view.

The [build](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) method of the \_GeneratePageState class, which is the state associated with the GeneratePage, constructs the user interface for this page. It returns a Row widget with two Expanded widgets as children. The first Expanded widget contains the file history view, and the second one contains the generate view.

The buildFileHistory method in the \_GeneratePageState class builds the file history view. It returns a Card widget that contains a Column widget. The Column widget has a Text widget for the title and an Expanded widget that contains a GeneratorHistory widget. The GeneratorHistory widget displays a list of previously generated files.

The GeneratorHistory widget is defined in [lib/screens/3\_GeneratePage/generator\_history.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). It takes a callback function onFileImported as a parameter, which is called when a file is imported.

The buildGenerateView method, which builds the generate view, constructs the user interface for generating Excel files and shows a summary of key details.

The ExcelWriter class in [lib/utils/excel\_writer.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is used for writing the data stored in the CellMapping class in [lib/models/cell\_mapping.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) using the Cell class in [lib/models/cell.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).

For accurately writing the formulas and the values to the excel file, there aren’t any suitable libraries available in dart, and so, a plugin ‘excel-writer’ ([github.com/jeryjs/Excel-Writer](https://github.com/jeryjs/Excel-Writer)) written in golang specifically for this app is used by compiling it into an executable (.exe) and running it in a separate process.

## Models

### AnalysedData

The file [lib/models/analysed\_data.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) defines a class AnalysedData that represents the analyzed data of a course.

The AnalysedData class has the following properties:

* component: Represents the component of the course.
* testNo: Represents the test number.
* courseName: Represents the name of the course.
* courseCode: Represents the code of the course.
* semester: Represents the semester.
* qpPattern: Represents the question paper pattern.
* error: Represents any error that occurred during the analysis.

The class provides the following methods:

* AnalysedData.fromJson(Map<String, dynamic> json): A factory constructor that creates an AnalysedData object from a JSON map.
* AnalysedData.empty(): A factory constructor that creates an empty AnalysedData object.
* isEmpty: A getter that checks if the AnalysedData object is empty.
* isNotEmpty: A getter that checks if the AnalysedData object is not empty.
* toJson(): A method that converts the AnalysedData object to a JSON string.

### CellMapping

The file [lib/models/cell\_mapping.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) defines a class CellMapping represents the cell mappings and holds other excel file related data:

The CellMapping  class has the following methods & properties:

* **detailsMapping**: This getter returns the details mapping stored in cellMappings[Mapping.details].
* **setDetailsMapping()**: This method sets the details mapping with the provided controllers. It also sets the courseCode property.
* **componentsMapping**: This getter returns the components mapping stored in cellMappings[Mapping.components].
* **setComponentsMapping()**: This method sets the components mapping with the provided components.
* **weightageMapping**: This getter returns the weightage mapping stored in cellMappings[Mapping.weightage].
* **setWeightageMapping()**: This method sets the weightage mapping with the provided controllers.
* **getStartCol()**: This method returns the excel style column name for the given column number.
* **generateOperations()**: This method generates a list of operations based on the cellMappings.

### Cell

The Cell class in [lib/models/cell.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) is a class that represents the cell:key mapping for each detail in the excel sheet.

The Cell class has static properties that represent different types of cells. For example, Cell.details.name, Cell.details.courseName, Cell.details.courseCode, etc. represent different cells in the "details" section of a spreadsheet. Similarly, Cell.weightage.coTarget, Cell.weightage.cos, etc. represent different cells in the "weightage" section of a spreadsheet.

The Cell class is used in the CellMapping class to map the values of different cells in a spreadsheet to the corresponding fields in the application. For example, in the setDetailsMapping method of the CellMapping class, the controllers parameter contains the values of the different fields in the application which are mapped to the corresponding cells in the spreadsheet using the properties of the Cell class.

In the setWeightageMapping method of the CellMapping class, the ctrl parameter is a list of TextEditingController objects that contain the values of the different weightages in the application. These values are then mapped to the corresponding cells in the spreadsheet using the properties of the Cell class.

In summary, the Cell class in [lib/models/cell.dart](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) provides a way to represent different types of cells in a spreadsheet and is used to map the values of these cells to the corresponding fields in the application.

## Assets

The [assets/](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) directory in contains files that are used by the application for its functioning. Here's a breakdown of each file and its purpose:

1. [assets/api/google.key](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): This file contains the API key for accessing Google’s Gemini AI used in QpAnalyser. To keep this file secure, it’s not shown in github repository.
2. assets/app\_icon.webp: This is the icon for the application.
3. [assets/excel-writer.exe](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): This is the executable file is used to write data to Excel files. It’s a compiled version of the excel-writer program written in go specifically for this app (github.com/jeryjs/Excel-Writer).
4. [assets/input.xlsx](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): This Excel file serves as an template for all excel files generated. The application reads data from this file for processing.
5. [assets/response\_format.json](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): This JSON file defines the format of the responses that your application receives from Gemini AI. It’s only for reference and isn’t actually involved in the functioning of the app.
6. [assets/samples/](vscode-file://vscode-app/z:/DO_NOT_TOUCH/Applications/VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): This directory holds sample images for training the Gemini AI. Multiple samples are provided for improving accuracy.
   * ./IA/1: Contains question paper from IA 5 (Sem 1)
   * ./IA/2: Contains question paper from IA 2 (Sem 3)

These files are bundled with the application during the build process and are available to the application at runtime. They are referenced in the application code using the paths specified in the AssetManifest.json file.

# [More detailed documentation is available at: co\_po\_generator - Dart API docs (jeryjs.github.io)](https://jeryjs.github.io/co_po_attainment_generator/)