**FormFiller Web Application**

The FormFiller project, powered by Python, is designed to streamline the process of filling various forms through automation. Leveraging the capabilities of Streamlit, this project offers a user-friendly interface for interacting with the system.

Table of Contents

[Running the Project: 1](#_Toc162618301)

[Project Structure: 1](#_Toc162618302)

[o app.py 2](#_Toc162618303)

[o template\base.py 2](#_Toc162618304)

[o pages\1\_Sign.py 2](#_Toc162618305)

[o pages\config.yaml 3](#_Toc162618306)

[o pages\credential.yaml 3](#_Toc162618307)

[o View: 3](#_Toc162618308)

[o View\analyzer.py 3](#_Toc162618309)

[o View\base.py 4](#_Toc162618310)

[o View\asker.py 4](#_Toc162618311)

[o streamlit: 5](#_Toc162618312)

[o config.toml: 5](#_Toc162618313)

[o secrets.toml: 5](#_Toc162618314)

# 

# Running the Project:

To initiate the project, execute the following command on the terminal:

**streamlit run app.py**

Project Structure:

Here I have explained the structure of the application along with its functions.

### app.py

#### Purpose:

This is the main application file responsible for handling file uploads and processing.

#### Methods:

* `on\_upload`: Handles the file uploading process. It saves the uploaded files, calls the `process\_files` function to process them, and displays the resulting questions.
* `process\_files`: Processes the uploaded files by determining their types and using the appropriate analyzer to extract questions. It then uses the `Asker` module to generate answers for these questions based on provided personal information.

### template\base.py

#### Purpose:

This directory provides the foundational interface structure for the project, encompassing both secret management and graphical user interface (GUI) components.

Handles the initialization and layout of the Streamlit template.

#### Methods:

* `template\_init`: Initializes the template by defining the home and sign pages.
* `template\_sidebar`: Sets up the sidebar layout.
* `confidential`: Checks the authentication status and hides or shows pages accordingly.

### pages\1\_Sign.py

#### Purpose:

Manages user authentication and redirects users based on their authentication status.

#### Functionality:

* It loads authentication configuration from `config.yaml`.
* Performs user authentication using provided credentials.
* Updates the authentication status and redirects the user to the appropriate page based on their authentication status.
* Saves the updated authentication status back to the configuration file.

### pages\config.yaml

#### Purpose:

Contains configuration settings for user authentication, such as cookie expiry, key, and preauthorized email addresses.

### pages\credential.yaml

#### Purpose:

Contains login credentials (username and password) for user authentication.

### View:

Within this directory, distinct functionalities are encapsulated in separate Python modules:

### View\analyzer.py

#### Purpose:

Contains classes and methods for analyzing different types of files to extract questions.

#### Classes:

* `Analyzer`: Base class for file analyzers.
* `ExcelAnalyzer`, `WordAnalyzer`, `PdfAnalyzer`: Subclasses of `Analyzer` specific to Excel, Word, and PDF files, respectively.

#### Functionality:

Each analyzer class contains methods for extracting and analyzing questions from the respective file types.

### View\base.py

#### Purpose:

Contains utility functions for separating files by extensions, as well as coordinating analysis and querying processes using the provided classes, interacting with the `Asker` module.

#### Methods:

* `is\_safe`: Checks if the user is authenticated.
* `process\_files`: Processes uploaded files by using appropriate analyzers and generates answers using the `Asker` module.

### View\asker.py

#### Purpose:

Handles interactions with OpenAI's GPT model to ask questions and receive answers.

#### Class:

`Asker`: Class for asking questions and receiving answers using OpenAI's GPT model.

#### Methods:

* `change\_info\_for\_person`: Asks questions and changes information for a person based on provided answers.
* `ask\_one\_person`: Asks questions to one person based on provided questions and person description.
* `ask`: Orchestrates the process of obtaining form structure, querying the GPT model, and aggregating responses.

### streamlit:

This directory houses configuration files defining interface settings and environment variables for the entire project.

### config.toml:

Specifies interface and environment configurations.

### secrets.toml:

Stores sensitive information such as the OpenAI API key.

## Appendix:

This is an example of text that is extracted and the answers generated. This method of the code gives better answers compared to the methods used in the algorithm.  
  
A screenshot of a computer

Description automatically generated

**Thank You**