File Uploader Module Documentation

Streamlit Interface for Financial Document Processing

Module Overview

This module (`file_uploader.py`) provides file loading functionality for the Gemini Pro Financial Decoder application. Built to work seamlessly with Streamlit, it handles the uploading and parsing of financial documents in multiple formats, ensuring data integrity and proper error handling.

What This Module Does

- Loads and parses financial documents from uploaded files
- Supports multiple file formats:
  - CSV files (`.csv`)
  - Excel files (`.xlsx`, `.xls`)
- Performs data validation and cleaning
- Handles file encoding and format detection automatically
- Provides comprehensive error handling with user-friendly messages
- Strips whitespace from column headers for consistency

Dependencies

This module depends on:

- **pandas** - for data manipulation and file parsing
- **streamlit** - for error messaging and UI integration
- **io** - for handling file streams and byte operations

Core Function: load_file(file)

```
def load_file(file):
    """Load and parse uploaded financial documents"""
```

Parameters

- **file**: A file-like object uploaded through Streamlit's file uploader component

Returns

- **pandas.DataFrame**: Successfully parsed data with cleaned column headers
- **None**: When file is empty, unsupported, or parsing fails

Functionality

- **File Validation**: Checks if file exists and is not None
- **Format Detection**: Automatically detects CSV and Excel formats based on file extension
- **Encoding Handling**: Properly decodes UTF-8 encoded CSV files
- **Data Cleaning**: Strips whitespace from column headers
- **Empty File Detection**: Warns users when uploaded files contain no data
- **Error Handling**: Catches and reports parsing errors with descriptive messages

Error Handling

The module implements robust error handling:

- **Unsupported Format**: Displays error for files that aren't CSV or Excel
- **Empty File Warning**: Alerts users when files contain no data
- **Parsing Errors**: Catches exceptions during file reading and provides detailed error messages
- **File Reset**: Automatically resets file pointer after reading content

Usage Example

```
import streamlit as st from file_uploader import
load_file # In your Streamlit app uploaded_file =
st.file_uploader("Upload Financial Document", type=
['csv', 'xlsx', 'xls']) if uploaded_file is not None: df
= load_file(uploaded_file) if df is not None:
st.success("File loaded successfully!") st.dataframe(df)
```

File Format Support

CSV Files

- Handles UTF-8 encoding automatically
- Parses comma-separated values
- Maintains data types where possible

Excel Files

- Supports both `.xlsx` and `.xls` formats
- Reads the first sheet by default
- Preserves Excel formatting and data types

Data Processing Features

- **Column Header Cleaning**: Removes leading/trailing whitespace from all column names

- **File Pointer Management**: Resets file position after reading to prevent issues
- **Memory Efficient**: Uses appropriate I/O streams for different file types

Error Messages

The module provides clear, actionable error messages:

- `"Unsupported format"` - When file type is not supported
- `"File is empty"` - When uploaded file contains no data
- `"File load error: {error_details}"` - When parsing fails with specific error information

Integration with Main Application

This module is designed to work seamlessly with the main Streamlit application:

- Called from `app.py` when users upload financial documents
- Returns standardized DataFrame format for consistent processing
- Provides user feedback through Streamlit's messaging system

📋 Module Source Code

```
import pandas as pd import streamlit as st import io
def load_file(file): if file is None: return None
try: content = file.read() file.seek(0) if
file.name.endswith('.csv'): df =
pd.read_csv(io.StringIO(content.decode('utf-8')))
elif file.name.endswith(('.xlsx', '.xls')): df =
pd.read_excel(io.BytesIO(content)) else:
st.error("Unsupported format") return None if
df.empty: st.warning("File is empty") return None
df.columns = df.columns.str.strip() return df except
```

```
Exception as e: st.error(f"File load error:
{str(e)}") return None
```

Contribution

This `file_uploader.py` module has been developed as a standalone utility to handle all file input operations for the financial analysis application. It maintains separation of concerns by focusing solely on file loading and parsing, while delegating data processing and analysis to other modules.

The module ensures data integrity and provides a reliable foundation for the entire financial analysis workflow, making it easy to extend support for additional file formats in the future.

Generated for Gemini Pro Financial Decoder | File Uploader Module Documentation