# Runbook for DeepFin Email Classification App

## Overview

This application classifies emails (including attachments) of the format PDF and EML files using DeepSeek AI model - **deepseek-r1:14b**. It consists of multiple Python scripts, including `app.py` for the Streamlit interface and `classifier.py` for text classification.

## Prerequisites

- Ensure Python is installed (>= 3.x).
- Install required dependencies using:`pip install -r requirements.txt`
- Set up and run Ollama before starting the application.

## File Structure

```
app.py              # Main application script (Streamlit UI)
classifier.py        # Text classification logic
requirements.txt       # Python dependencies
categories.txt        # Defines top-level categories
ruleset.json         # Maps categories to subcategory files
resources/           # Contains necessary model files or configurations
temp/               # Temporary storage for processed files
```

## Initial Setup Steps

### Define Categories in categories.txt

Create a file named `categories.txt` in the project directory to list the top-level categories. Example content:

```
Categories:
1. Category Name: Money Movement - Inbound
```

    * Description: This refers to any transaction where money flows into the bank from an external source. It includes loan repayments, interest payments, fees collected, or any deposits received from customers or third parties. These transactions increase the bank’s balance and are typically marked as credits.
    * Sample Email: Receive $14,000 inbound for Deal MNO on 03/25/2025, account 77889.

## Create ruleset.json

Create a file named ruleset.json in the project directory to map each category to its subcategory file. Example content:

```
{
    "Money Movement - Inbound": "resources/sub-category-money-movement-inbound.txt"
}
```

## Create Subcategory File (sub-category-money-movement-inbound.txt)

Inside the resources/ directory, create a file named sub-category-money-movement-inbound.txt to define subcategories for "Money Movement - Inbound". Example content:

```
Categories:
1. Category Name: Principal
    * Description: This is when someone pays back just the original chunk of cash they borrowed. No extras, just the straight-up loan amount coming back to us.
    * Sample Email: Receive $15,000 principal payment for Deal NOP on 03/18/2025, account 98765.
    * Sample Email: Record $12,500 principal for Deal ZAB on 03/27/2025, account 22334.
```

# Setting Up Ollama

- Install Ollama following the official documentation.
- Start Ollama before running the application:ollama start
- Ensure Ollama is running properly before executing the application.

## Running the Application

1. Navigate to the project directory:`cd /path/to/project`

2. Start the Streamlit app:`streamlit run app.py`

3. Upload a PDF or EML file in the UI and view classification results.

## Running Unit Tests

To execute unit tests and generate coverage reports, run the following command:

```
python3 -m pytest --html=reports/unit_test_report.html --self-contained-html --cov=app --cov=classifier --cov-report=html:reports/coverage_report --cov-report=term
```

This generates an HTML unit test report in `reports/unit_test_report.html`. Coverage reports are saved in `reports/coverage_report` (HTML) and displayed in the terminal.

## Debugging & Logs

- Check the terminal for errors.
- Use `print()` or logging in `app.py` and `classifier.py` for debugging.
- Ensure Ollama is running properly if classification is not working.

## Updating Dependencies

To update dependencies, modify `requirements.txt` and run:

```
pip install -r requirements.txt --upgrade
```

## Additional Notes

- Ensure `resources/` contains necessary models and subcategory files (e.g., `sub-category-money-movement-inbound.txt`).

- Temporary files are stored in `temp/` and may be cleaned periodically.
- For further assistance, refer to internal documentation or contact the developer.

There is also a .MD file of the runback which shows in better format