Using pdfrw and num2words libraries to generate pdf files

1. Quick summary of the tech.

This is a **Flask-based REST API** that:

- Accepts structured JSON data (about a payment, client, and vehicle),
- Fills out a PDF receipt template with that data,
- Returns the filled PDF as a downloadable file.

Pdfrw

- **What**: A pure Python library to read/write/edit PDF files.
- **Used for**: Loading your PDF receipt template and filling form fields (AcroForms).
- **Real-life use**: Generating auto-filled forms like receipts, invoices, or official documents (e.g., government permit templates).

Num2words

- **What**: Converts numbers to words (e.g., `123 → "one hundred twenty-three"`).
- **Used for**: Writing the payment amount in words on the receipt.
- **Real-life use**: Used in financial documents, invoices, and cheques for clarity and fraud prevention.

Real-Life Use Case Example

Imagine you're running a **car dealership system**:

- A customer makes a payment.
- The system hits this API with payment + client + car data.
- The API returns a **ready-to-print receipt** — filled with:
    - Customer name & ID
    - Car make/model & registration
    - Amount paid (both numeric and in words)
    - Payment method (MPESA, bank, etc.)
- The PDF is either downloaded automatically or stored.

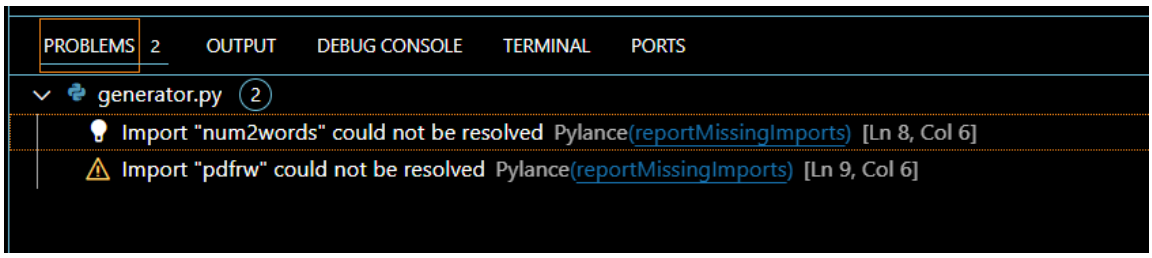
2. System Requirements

OS: WLS or Linux.

Editors: Visual Studio Code.

Packages: pdfrw, num2words (pip).

3. Installations and Setup Instructions



That's the first error message I got, the image below showcases on how to counter it:

```
(BorneLabsBackend) lance@DYLAN-15:~/Development/code/Moringa-Gen-AI$ pipenv install num2words
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instea
ct. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and create its own instead. You can set P
 warning.
Installing num2words...
Resolving num2words...
Added num2words to Pipfile's [packages] ...
✔ Installation Succeeded
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Building requirements...
Resolving dependencies...
✔ Success!
Locking [dev-packages] dependencies...
Updated Pipfile.lock (10f8517947deb835bcf0a762283cf3bcff1a36eec726585c1bc680312e8341b7)!
Installing dependencies from Pipfile.lock (8341b7)...
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
(BorneLabsBackend) lance@DYLAN-15:~/Development/code/Moringa-Gen-AI$ pipenv install pdfrw
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instea
ct. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and create its own instead. You can set P
 warning.
Installing pdfrw...
Resolving pdfrw...
Added pdfrw to Pipfile's [packages] ...
```

And later run the file, in my case it is the generator.py:

```
(BorneLabsBackend) lance@DYLAN-15:~/Development/code/Moringa-Gen-AI$ python generator.py
 * Serving Flask app 'generator'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5900
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 271-916-246
```

4. Working Example

```
# generates a random 6 digit number with current year and month

def generate_otp_with_date():

    otp = random.randint(100000, 999999)

    now = datetime.now()

    return f"{otp}/{now.strftime('%Y-%m')}"



class ReceiptGenerator(Resource):
```

```
    def post(self):
        try:
            data = request.get_json() # Get JSON data from the
request body


            # Data unpacked from request dictionary
            payment = data['payment']
            vehicle = data['vehicle']
            client = data['client']


            pdf_bytes = self.generate_payment_pdf(payment, vehicle,
client)


            # the send_file function sends the PDF file as a
response to be viewed
            return send_file(
                io.BytesIO(pdf_bytes),
                download_name=f"Payment_{payment['id']}.pdf",
                mimetype='application/pdf',
                as_attachment=True # Downloads the file instead of
displaying it in the browser
            )
        except Exception as e:
```

```
            return jsonify({'error': f'Failed to generate payment
receipt: {str(e)}'}), 500



    # The generate PDF function

    def generate_payment_pdf(self, payment, vehicle, client):

        template_path = "static/entry/RIFT-CARS-RECEIPT.pdf"

        template = PdfReader(template_path)



        # function to convert amount to words

        def amount_to_words(amount):

            shillings = int(amount)

            cents = int(round((amount - shillings) * 100))

            words = num2words(shillings, lang='en') + " Kenyan
shillings"

            if cents:

                words += f" and {num2words(cents, lang='en')} cents"

            return words.upper()



        if payment['payment_method'] == 'mpesa':

            description = f"MPESA Paybill No
{payment.get('mpesa_account_number', '')}"

        elif payment['payment_method'] == 'bank':

            description = f"Bank Account No
{payment.get('bank_account_number', '')}"

        else:
```

```
            description = f"Payment for {vehicle['make']}
{vehicle['model']}"



        field_values = {

            'PaymentID': generate_otp_with_date(),

            'TransactionNo': payment['transaction_number'],

            'Amount': f"{payment['amount']:.2f}",

            'AmountInWords': amount_to_words(payment['amount']),

            'PaymentMode': payment['payment_method'].capitalize(),

            'CarDesc': f"{vehicle['make']} {vehicle['model']}",

            'Description': description,

            'Authority': payment['authorized_by'],

            'ClientName': client['name'],

            'ClientID': client['id_number'],

            'CarReg': vehicle.get('registration_number', ''),

            'Date': payment['payment_date']

        }



        # Set NeedAppearances flag - Ensures field values are
visible in PDF viewers

        if not template.Root.AcroForm:

            template.Root.AcroForm = PdfDict()


template.Root.AcroForm.update(PdfDict(NeedAppearances=PdfObject('tru
e')))
```

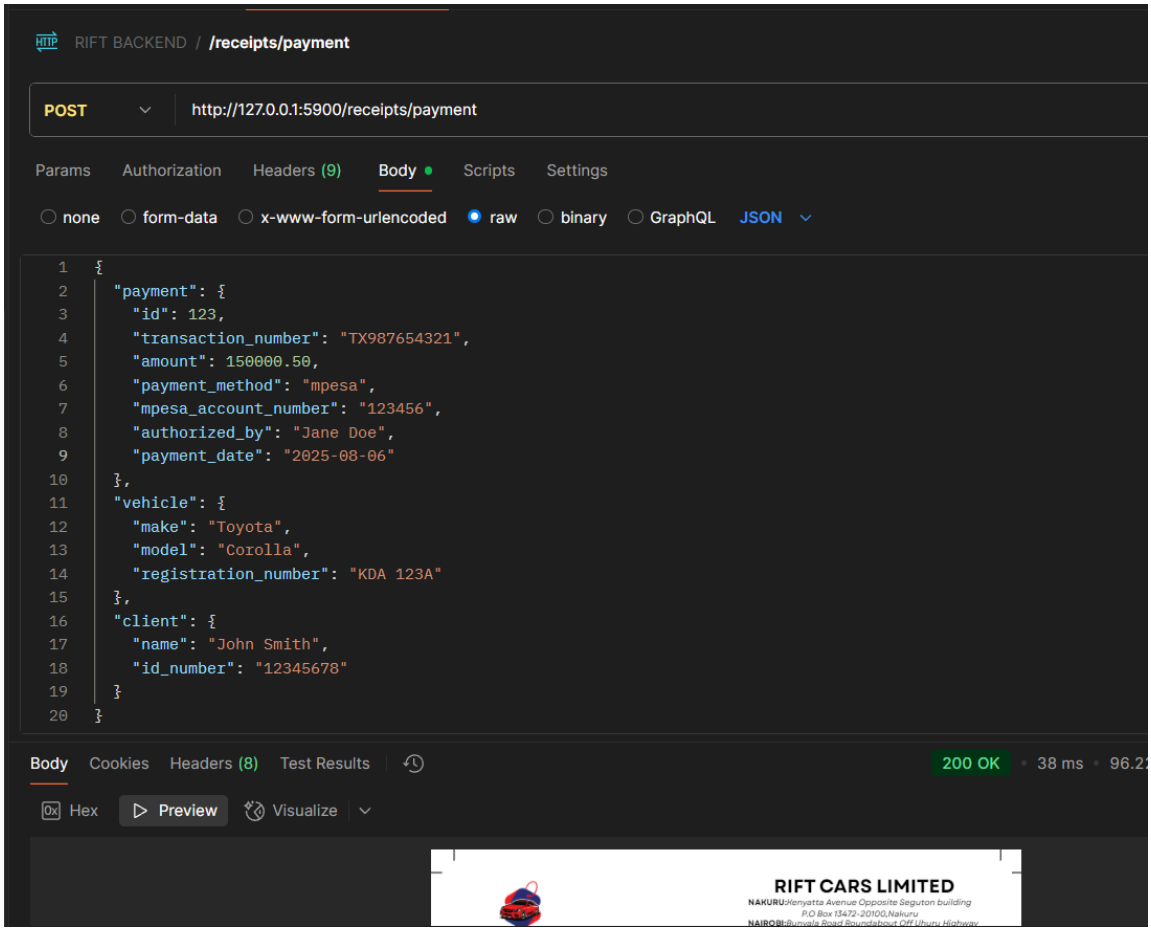
```
        for page in template.pages:
            annotations = page.Annots
            if annotations:
                for annot in annotations:
                    key = annot.T
                    if key:
                        field_name = key.to_unicode().strip()
                        if field_name in field_values:
                            value = field_values[field_name]
                            annot.update(
                                PdfDict(
                                    V=PdfString.encode(str(value)),
                                    Ff=1,  # Makes the file
read-only
                                )
                            )


            # Writes and saves the modified PDF into memory.
            output_buffer = io.BytesIO()
            PdfWriter().write(output_buffer, template)
            output_buffer.seek(0)
            return output_buffer.read()
```

```
api.add_resource(ReceiptGenerator, '/receipts/payment')
```

And paste this request in postman:

```
{
  "payment": {
    "id": 123,
    "transaction_number": "TX987654321",
    "amount": 150000.50,
    "payment_method": "mpesa",
    "mpesa_account_number": "123456",
    "authorized_by": "Jane Doe",
    "payment_date": "2025-08-06"
  },
  "vehicle": {
    "make": "Toyota",
    "model": "Corolla",
    "registration_number": "KDA 123A"
  },
  "client": {
    "name": "John Smith",
    "id_number": "12345678"
  }
}
```

The below image showcases it better on how to run in postman:

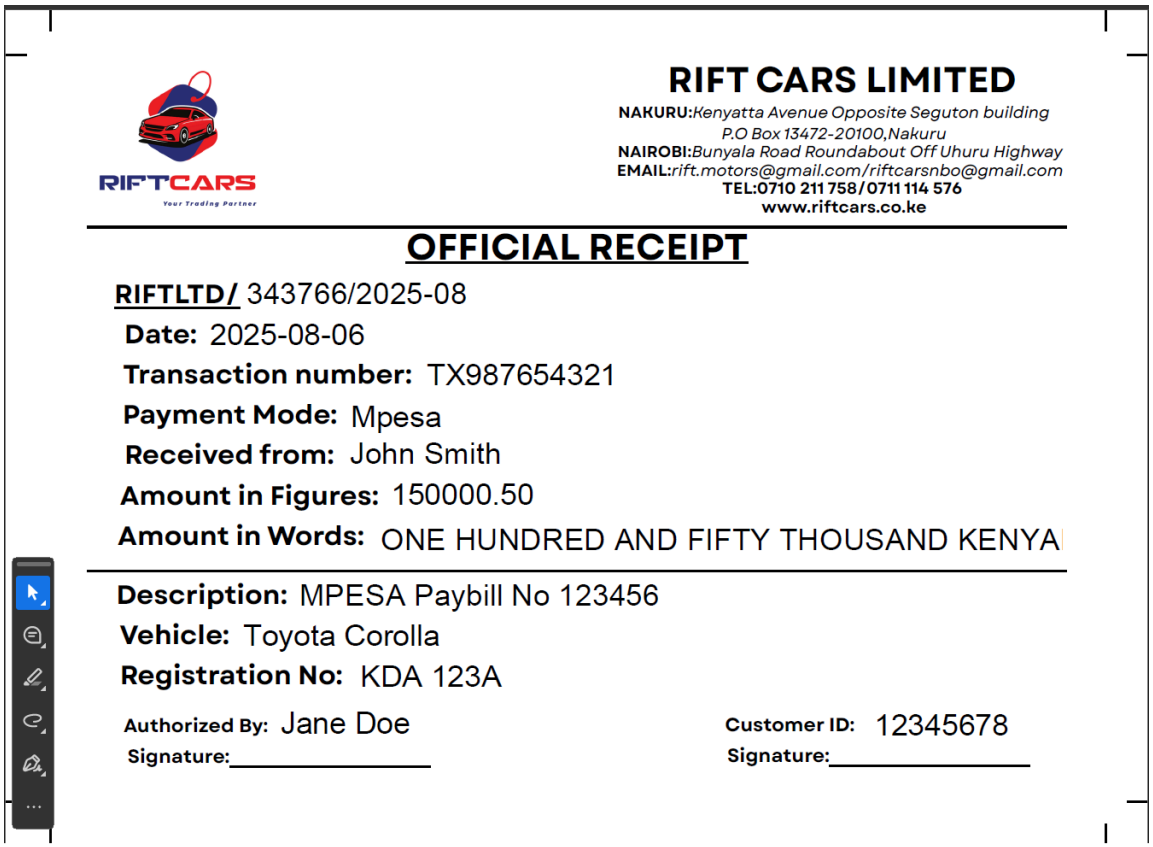


To download the generated document from postman:

After making the POST request:

1. Go to the **"Body"** tab of the response.
2. Click **"Save Response"** (small download icon).
3. Choose where to store it — done!

Output is a pdf file with filled data in it like below:



RIFTCARS
Your Trading Partner

RIFT CARS LIMITED

NAKURU: *Kenyatta Avenue Opposite Seguton building*
P.O Box 13472-20100,Nakuru
NAIROBI: *Bunyala Road Roundabout Off Uhuru Highway*
EMAIL: *rift.motors@gmail.com/riftcarsnbo@gmail.com*
TEL:0710 211 758 / 0711 114 576
www.riftcars.co.ke

OFFICIAL RECEIPT

RIFTLTD/ 343766/2025-08
Date: 2025-08-06
Transaction number: TX987654321
Payment Mode: Mpesa
Received from: John Smith
Amount in Figures: 150000.50
Amount in Words: ONE HUNDRED AND FIFTY THOUSAND KENYA

Description: MPESA Paybill No 123456
Vehicle: Toyota Corolla
Registration No: KDA 123A

Authorized By: Jane Doe
Signature:_____

Customer ID: 12345678
Signature:_____

5. AI Prompt Journal

Here is the link to my conversation with chatGPT:

https://chatgpt.com/share/6893a82e-d7b8-8007-92fc-8c3b366dfe52

6. Common Issues & Fixes

1. An edge case to consider if the pdf attachment should be force downloaded or just be viewed in frontend use/browser. Handled this by setting as_attachment=True to force download and vice versa to be only as viewed.

2. The pdf fields were not being filled, hence needed to have needappearances flag to be set to True so as to be compatible and viewed by all PDF Viewers. NeedAppearances=PdfObject('true'))

3. The PDF generated was editable at first hence needed to specify Ff=1 in the PdfDict class to counter the problem (to be non-editable).

4. Noted that maybe not all people may know that one can download a media type response(file/pdf/image) from postman hence can state it here:

After making the `POST` request:

1. Go to the **"Body"** tab of the response.
2. Click **"Save Response"** (small download icon).
3. Choose where to store it — done!

7. References

I learned more on pdfrw from this link in stack overflow (https://stackoverflow.com/questions/47288578/pdf-form-filled-with-pypdf2-does-not-show-in-print) and documentation (https://github.com/pmaupin/pdfrw)