```
!pip install fastapi uvicorn jinja2 matplotlib python-multipart pdfkit pydantic pyngrok
!apt-get install -y wkhtmltopdf
```

```
Requirement already satisfied: fastapi in /usr/local/lib/python3.12/dist-packages (0.118.3)
Requirement already satisfied: uvicorn in /usr/local/lib/python3.12/dist-packages (0.38.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (3.1.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: python-multipart in /usr/local/lib/python3.12/dist-packages (0.0.20)
Collecting pdfkit
  Downloading pdfkit-1.0.0-py3-none-any.whl.metadata (9.3 kB)
Requirement already satisfied: pydantic in /usr/local/lib/python3.12/dist-packages (2.12.3)
Collecting pyngrok
  Downloading pyngrok-7.5.0-py3-none-any.whl.metadata (8.1 kB)
Requirement already satisfied: starlette<0.49.0,>=0.40.0 in /usr/local/lib/python3.12/dist-packages (from fastapi) (0.48.0
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.12/dist-packages (from fastapi) (4.15.0)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.12/dist-packages (from uvicorn) (8.3.1)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.12/dist-packages (from uvicorn) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2) (3.0.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.po
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic) (0.4.2)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib)
Requirement already satisfied: anyio<5,>=3.6.2 in /usr/local/lib/python3.12/dist-packages (from starlette<0.49.0,>=0.40.0-
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-packages (from anyio<5,>=3.6.2->starlette<0.49.
Downloading pdfkit-1.0.0-py3-none-any.whl (12 kB)
Downloading pyngrok-7.5.0-py3-none-any.whl (24 kB)
Installing collected packages: pdfkit, pyngrok
Successfully installed pdfkit-1.0.0 pyngrok-7.5.0
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core avahi-daemon geoclue-2.0 glib-networking glib-networking-common
  glib-networking-services gsettings-desktop-schemas gstreamer1.0-plugins-base
  iio-sensor-proxy libatk-bridge2.0-0 libatk1.0-0 libatk1.0-data libatspi2.0-0
  libavahi-core7 libavahi-glib1 libcdparanoia0 libdaemon0
  libdouble-conversion3 libevdev2 libgstreamer-plugins-base1.0-0 libgtk-3-0
  libgtk-3-bin libgtk-3-common libgudev-1.0-0 libhyphen0 libinput-bin
  libinput10 libjson-glib-1.0-0 libjson-glib-1.0-common libmbim-glib4
  libmbim-proxy libmd4c0 libmm-glib0 libmtdev1 libnl-genl-3-200 libnotify4
  libnss-mdns liborc-0.4-0 libproxy1v5 libqmi-glib5 libqmi-proxy libqt5core5a
  libqt5dbus5 libqt5gui5 libqt5network5 libqt5positioning5 libqt5printsupport5
  libqt5qml5 libqt5qmlmodels5 libqt5quick5 libqt5sensors5 libqt5svg5
  libqt5webchannel5 libqt5webkit5 libqt5widgets5 librsvg2-common libsoup2.4-1
  libsoup2.4-common libudev1 libvisual-0.4-0 libwacom-bin libwacom-common
  libwacom9 libwoff1 libxcb-icccm4 libxcb-image0 libxcb-keysyms1
  libxcb-render-util0 libxcb-util1 libxcb-xinerama0 libxcb-xinput0 libxcb-xkb1
  libxcomposite1 libxkbcommon-x11-0 libxtst6 modemmanager
  qt5-gtk-platformtheme qttranslations5-l10n session-migration
  systemd-hwe-hwdb udev usb-modeswitch usb-modeswitch-data wpasupplicant
```

```
from pyngrok import ngrok

ngrok.set_auth_token("351HKLlWxcKdySlodvXv1tdtn4n_7nd2uGd4azL8iLUx")
ngrok.kill()

public_url = ngrok.connect(8000)
public_url
```

```
<NgrokTunnel: "https://unprovokable-medianly-clayton.ngrok-free.dev" -> "http://localhost:8000">
```

```
%%writefile app.py
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse, FileResponse
from pydantic import BaseModel, Field
from typing import List, Optional
from datetime import datetime, date
import uuid, pdfkit, os


# FastAPI
app = FastAPI(title="GramIQ Report Generator")
```

```
# Pydantic Models

class MoneyRow(BaseModel):
    category: str = Field(..., min_length=1)
    amount: float = Field(..., ge=0)
    date: date
    description: Optional[str] = ""

class FarmerCrop(BaseModel):
    farmer_name: str
    crop_name: str
    season: str
    total_acres: float
    date_of_sowing: Optional[date] = None
    date_of_harvest: Optional[date] = None
    location: Optional[str] = None


# Service Layer (Summaries / Ledger / Aggregates)

from decimal import Decimal, ROUND_HALF_UP

def _to_decimal(x):
    return Decimal(str(x)) if x is not None else Decimal("0")

def compute_totals(income_rows, expense_rows):
    ti = sum(_to_decimal(r["amount"]) for r in income_rows) if income_rows else Decimal("0")
    te = sum(_to_decimal(r["amount"]) for r in expense_rows) if expense_rows else Decimal("0")
    net = ti - te
    return {
        "total_income": ti.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP),
        "total_expense": te.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP),
        "net_income": net.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP),
    }

def cost_per_acre(total_expense: Decimal, acres: float):
    if acres is None or acres == 0:
        return None
    cpa = total_expense / _to_decimal(acres)
    return cpa.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP)

def aggregate_by_category(rows):
    agg = {}
    for r in rows:
        cat = r["category"]
        amt = _to_decimal(r["amount"])
        agg[cat] = agg.get(cat, Decimal("0")) + amt
    return sorted([(k, v) for k, v in agg.items()], key=lambda x: x[1], reverse=True)

def generate_ledger(income_rows, expense_rows):
    ledger = []
    for r in income_rows:
        ledger.append({
            "date": r.get("date"),
            "particulars": r.get("category"),
            "type": "Income",
            "description": r.get("description", ""),
            "amount": _to_decimal(r.get("amount")),
        })
    for r in expense_rows:
        ledger.append({
            "date": r.get("date"),
            "particulars": r.get("category"),
            "type": "Expense",
            "description": r.get("description", ""),
            "amount": _to_decimal(r.get("amount")),
        })

    ledger.sort(key=lambda x: (x["date"] is None, x["date"] or ""))
    return ledger

# Charting (matplotlib → base64)
import io, base64
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

def _b64(buf: io.BytesIO):
    buf.seek(0)
    return base64.b64encode(buf.read()).decode("ascii")

def chart_income_vs_expense(total_income, total_expense):
```

```
    fig, ax = plt.subplots(figsize=(4,3))
    ax.bar(["Income", "Expense"], [float(total_income), float(total_expense)])
    ax.set_title("Income vs Expense")
    plt.tight_layout()
    buf = io.BytesIO()
    fig.savefig(buf, format="png", bbox_inches="tight")
    plt.close(fig)
    return _b64(buf)

def chart_expense_distribution(expense_by_cat):
    if not expense_by_cat:
        return ""
    labels = [cat for cat,_ in expense_by_cat]
    sizes  = [float(val) for _,val in expense_by_cat]
    fig, ax = plt.subplots(figsize=(4,3))
    ax.pie(sizes, labels=labels, autopct="%1.1f%%")
    ax.set_title("Expense Distribution")
    ax.axis("equal")
    plt.tight_layout()
    buf = io.BytesIO()
    fig.savefig(buf, format="png", bbox_inches="tight")
    plt.close(fig)
    return _b64(buf)

# HTML Form
HTML_FORM = """
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>GramIQ Form</title>
<style>input{padding:6px;margin:3px;} .row{margin-bottom:6px}</style>
</head>
<body>
<h2>GramIQ Report Form</h2>

<form method="post" action="/generate">
  <input name="farmer_name" placeholder="Farmer name" required>
  <input name="crop_name" placeholder="Crop name" required>
  <input name="season" placeholder="Season" required>
  <input name="total_acres" placeholder="Total acres" type="number" step="0.01" required>
  <input name="date_of_sowing" type="date">
  <input name="date_of_harvest" type="date">
  <input name="location" placeholder="Location">

  <h4>Income</h4>
  <div id="income-list">
    <div class="row">
      <input name="income_category" placeholder="Category" required>
      <input name="income_amount" type="number" step="0.01" required>
      <input name="income_date" type="date" required>
    </div>
  </div>
  <button type="button" onclick="addIncome()">+ Add Income</button>

  <h4>Expenses</h4>
  <div id="expense-list">
    <div class="row">
      <input name="expense_category" placeholder="Category" required>
      <input name="expense_amount" type="number" step="0.01" required>
      <input name="expense_date" type="date" required>
    </div>
  </div>
  <button type="button" onclick="addExpense()">+ Add Expense</button>

  <div><button type="submit">Generate PDF</button></div>
</form>

<script>
function addIncome(){
  const d=document.createElement('div'); d.className='row';
  d.innerHTML=`<input name="income_category" placeholder="Category" required>
               <input name="income_amount" type="number" step="0.01" required>
               <input name="income_date" type="date" required>`;
  document.getElementById('income-list').appendChild(d);
}
function addExpense(){
  const d=document.createElement('div'); d.className='row';
  d.innerHTML=`<input name="expense_category" placeholder="Category" required>
               <input name="expense_amount" type="number" step="0.01" required>
               <input name="expense_date" type="date" required>`;
  document.getElementById('expense-list').appendChild(d);
}
```

```
}
</script>

</body>
</html>
"""

# Helper to collect repeated rows
def _collect(form, prefix):
    cats = form.getlist(f"{prefix}_category")
    amts = form.getlist(f"{prefix}_amount")
    dates = form.getlist(f"{prefix}_date")
    rows=[]
    for i in range(len(cats)):
        rows.append({
            "category": cats[i],
            "amount": float(amts[i]),
            "date": dates[i],
        })
    return rows

# Attempt to load logo base64 (optional)
logo_b64 = None
if os.path.exists("gramiq_logo.png"):
    try:
        with open("gramiq_logo.png", "rb") as f:
            logo_b64 = base64.b64encode(f.read()).decode("ascii")
    except Exception:
        logo_b64 = None

# Routes
@app.get("/", response_class=HTMLResponse)
def home():
    return HTML_FORM

@app.post("/generate")
async def generate(request: Request):
    form = await request.form()

    meta = {
        "farmer_name": form.get("farmer_name"),
        "crop_name": form.get("crop_name"),
        "season": form.get("season"),
        "total_acres": float(form.get("total_acres") or 0),
        "date_of_sowing": form.get("date_of_sowing"),
        "date_of_harvest": form.get("date_of_harvest"),
        "location": form.get("location"),
    }

    income_rows  = _collect(form, "income")
    expense_rows = _collect(form, "expense")

    totals = compute_totals(income_rows, expense_rows)
    cpa    = cost_per_acre(totals["total_expense"], meta["total_acres"])
    exp_cat = aggregate_by_category(expense_rows)
    ledger  = generate_ledger(income_rows, expense_rows)

    chart_ie  = chart_income_vs_expense(totals["total_income"], totals["total_expense"])
    chart_pie = chart_expense_distribution(exp_cat)

    fname = f"{uuid.uuid4()}.pdf"

    # Render Jinja2 template
    from jinja2 import Environment, FileSystemLoader
    env = Environment(loader=FileSystemLoader("templates"))
    tpl = env.get_template("report.html")

    report_title = f"{meta['crop_name']}_{meta['season']}_{datetime.utcnow().year}"
    generated_on = datetime.utcnow().strftime("%Y-%m-%d %H:%M UTC")

    charts_dict = {
        "income_expense": chart_ie,
        "expense_pie": chart_pie
    }

    html = tpl.render(
        meta=meta,
        report_title=report_title,
        generated_on=generated_on,
        totals=totals,
        cost_per_acre=cpa,
        charts=charts_dict,
        income_rows=income_rows,
```

```
        expense_rows=expense_rows,
        ledger=ledger,
        logo_base64=logo_b64
    )

    # generate PDF (pdfkit + wkhtmltopdf must be installed)
    pdfkit.from_string(html, fname)

    return FileResponse(fname, media_type="application/pdf", filename="gramiq_report.pdf")
```

```
Overwriting app.py
```

```
!pkill -f uvicorn || true

import subprocess, sys, time
process = subprocess.Popen(
    [sys.executable, "-m", "uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]
)

time.sleep(2)
print("Uvicorn server started on port 8000")
process
```

```
^C
Uvicorn server started on port 8000
<Popen: returncode: None args: ['/usr/bin/python3', '-m', 'uvicorn', 'app:ap...>
```

```
%%writefile templates/report.html
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
  body { font-family: Arial; font-size: 12px; }
  h1 { margin-bottom: 4px; }
  table { width:100%; border-collapse: collapse; margin-top:10px; }
  th, td { border:1px solid #ccc; padding:6px; }
  th { background:#f2f2f2; }
</style>
</head>
<body>

<div style="display:flex; align-items:center; border-bottom:1px solid #ccc; padding-bottom:10px; margin-bottom:15px;">
    {% if logo_base64 %}
        <img src="{{ logo_base64 }}" width="80" style="margin-right:12px;">
    {% endif %}

    <div>
        <h1 style="margin:0; padding:0;">{{ report_title }}</h1>

        <div style="font-size:12px; color:#333;">
            Farmer: {{ meta.farmer_name }} |
            Season: {{ meta.season }} |
            Acres: {{ meta.total_acres }}
        </div>

        <div style="font-size:11px; color:#555;">
            Generated: {{ generated_on }}
        </div>
    </div>
</div>

<h1>{{ report_title }}</h1>
<p>Farmer: {{ meta.farmer_name }} | Season: {{ meta.season }} | Acres: {{ meta.total_acres }}</p>
<p>Generated on: {{ generated_on }}</p>

<h3>Finance Summary</h3>
<ul>
  <li>Total Income: {{ totals.total_income }}</li>
  <li>Total Expense: {{ totals.total_expense }}</li>
  <li>Net Income: {{ totals.net_income }}</li>
  <li>Cost per acre: {{ cost_per_acre }}</li>
</ul>

<h3>Charts</h3>
<img src="{{ charts.income_expense }}" width="300">
<img src="{{ charts.expense_pie }}" width="300">

<h3>Income Breakdown</h3>
<table>
```

```
  <tr><th>Category</th><th>Amount</th><th>Date</th></tr>
  {% for r in income_rows %}
  <tr><td>{{ r.category }}</td><td>{{ r.amount }}</td><td>{{ r.date }}</td></tr>
  {% endfor %}
</table>

<h3>Expense Breakdown</h3>
<table>
  <tr><th>Category</th><th>Amount</th><th>Date</th></tr>
  {% for r in expense_rows %}
  <tr><td>{{ r.category }}</td><td>{{ r.amount }}</td><td>{{ r.date }}</td></tr>
  {% endfor %}
</table>

<h3>Ledger</h3>
<table>
  <tr><th>Date</th><th>Particulars</th><th>Type</th><th>Amount</th></tr>
  {% for row in ledger %}
  <tr><td>{{ row.date }}</td><td>{{ row.particulars }}</td><td>{{ row.type }}</td><td>{{ row.amount }}</td></tr>
  {% endfor %}
</table>

</body>

</html>
```

```
Overwriting templates/report.html
```

```
!ls
```

```
 app.py        sample_data                          templates
 __pycache__  'Screenshot 2025-12-11 224514.png'
```

```
import base64

with open("Screenshot 2025-12-11 224514.png", "rb") as f:
    logo_b64 = base64.b64encode(f.read()).decode("ascii")
```

```
!ls -1
```

```
app.py
__pycache__
sample_data
'Screenshot 2025-12-11 224514.png'
templates
```