**PROLOG CODE (tax\_advisor2.pl):**

% -----------------------------

% Dynamic predicates

:- dynamic tax\_slab/3.

:- dynamic income/1.

:- dynamic age/1.

:- dynamic regime/1.

:- dynamic deduction/2.

:- dynamic old\_regime\_tax/1.

:- dynamic new\_regime\_tax/1.

% -----------------------------

% Tax slabs FY 2024–25

% Old Regime

tax\_slab(old, 250000, 0).

tax\_slab(old, 500000, 0.05).

tax\_slab(old, 1000000, 0.2).

tax\_slab(old, 5000000, 0.3).

tax\_slab(old, 99999999, 0.3). % surcharge applies separately

% New Regime

tax\_slab(new, 300000, 0).

tax\_slab(new, 700000, 0.05).

tax\_slab(new, 1000000, 0.1).

tax\_slab(new, 1200000, 0.15).

tax\_slab(new, 1500000, 0.2).

tax\_slab(new, 5000000, 0.3).

tax\_slab(new, 99999999, 0.3). % surcharge applies separately

% -----------------------------

% Surcharge slabs

surcharge(\_, 5000000, 0.0).

surcharge(\_, 10000000, 0.10).

surcharge(\_, 20000000, 0.15).

surcharge(old, 50000000, 0.25).

surcharge(new, 999999999, 0.25).

surcharge(old,999999999, 0.37).

% -----------------------------

% Max deduction limits

max\_deduction('80C', 150000).

max\_deduction('80D', Max) :- age(A), (A >= 60 -> Max = 50000 ; Max = 25000).

max\_deduction('80CCD(1B)', 50000).

max\_deduction('EPF', 150000).

max\_deduction('Life Insurance', 150000).

max\_deduction('Standard', 50000).

max\_deduction('NPS', 50000).

% -----------------------------

% Utility Rules

unify\_example :-

X = tax(slab, rate),

writeln(X).

sum\_list\_custom([], 0).

sum\_list\_custom([H|T], Sum) :-

sum\_list\_custom(T, Rest),

Sum is H + Rest.

not\_claimed(Section) :-

\+ deduction(Section, \_).

first\_tax\_slab(Regime, Limit, Rate) :-

tax\_slab(Regime, Limit, Rate), !.

ask\_user\_section :-

write("Enter a section to check: "),

read(Sec),

( deduction(Sec, Amt) -> format("Claimed amount: ₹~w~n", [Amt])

; writeln("Not claimed.") ).

show\_text\_chart(OldTax, NewTax) :-

writeln("📊 [Chart will be displayed in Python]").

age\_based\_80d\_limit(Limit) :-

age(A),

( A >= 60 -> Limit = 50000 ; Limit = 25000 ).

total\_deductions(Sum) :-

findall(A, deduction(\_, A), L),

sum\_list\_custom(L, Partial),

max\_deduction('Standard', Std),

Sum is Partial + Std.

age\_based\_exemption(E) :-

age(A),

(A >= 60 -> E is 300000 ; E is 250000).

get\_sorted\_slabs(Regime, Sorted) :-

findall((Limit, Rate), tax\_slab(Regime, Limit, Rate), L),

sort(L, Sorted).

% -----------------------------

% Taxable Income

taxable\_income(old, TI) :-

income(I),

total\_deductions(D),

age\_based\_exemption(E),

Raw is I - D - E,

TI is max(0, Raw).

taxable\_income(new, TI) :-

income(I),

TI is I - 50000.

% -----------------------------

% Progressive Tax Computation

progressive\_tax(\_, \_, [], \_, 0).

progressive\_tax(Income, PrevLimit, [(Limit, Rate)|Rest], Acc, Tax) :-

Income =< Limit,

Portion is Income - PrevLimit,

Temp is Portion \* Rate,

Tax is Acc + Temp.

progressive\_tax(Income, PrevLimit, [(Limit, Rate)|Rest], Acc, Tax) :-

Income > Limit,

Portion is Limit - PrevLimit,

Temp is Portion \* Rate,

NewAcc is Acc + Temp,

progressive\_tax(Income, Limit, Rest, NewAcc, Tax).

% -----------------------------

%surcharge application

apply\_surcharge(Regime, BaseTax, FinalTax) :-

income(I),

findall((Limit, Rate), surcharge(Regime, Limit, Rate), Slabs),

sort(Slabs, Sorted),

find\_surcharge\_rate(I, Sorted, 0, SurchargeRate),

FinalTax is BaseTax \* (1 + SurchargeRate).

find\_surcharge\_rate(Income, [(Limit, Rate)|Rest], CurrentRate, FinalRate) :-

( Income =< Limit ->

FinalRate = Rate

;

find\_surcharge\_rate(Income, Rest, Rate, FinalRate)

).

find\_surcharge\_rate(\_, [], Rate, Rate).

% -----------------------------

% Compute Final Tax

compute\_tax(Regime, FinalTax) :-

taxable\_income(Regime, TI),

get\_sorted\_slabs(Regime, Slabs),

progressive\_tax(TI, 0, Slabs, 0, BaseTax),

apply\_surcharge(Regime, BaseTax, FinalTax).

% -----------------------------

% Regime Suggestion

suggest\_regime(BestRegime, OldTax, NewTax) :-

compute\_tax(old, OldTax),

compute\_tax(new, NewTax),

(OldTax < NewTax -> BestRegime = old ; BestRegime = new).

% -----------------------------

% Deduction Gap Logic

deduction\_gap('80C', Gap) :-

findall(Amount,

(member(Section, ['80C', 'EPF', 'LifeInsurance', 'NPS']),

deduction(Section, Amount)),

Amounts),

sum\_list\_custom(Amounts, Total),

Gap is 150000 - Total,

Gap > 0.

deduction\_gap('80D', Gap) :-

(deduction('80D', Used) -> true ; Used = 0),

max\_deduction('80D', Max),

Gap is Max - Used,

Gap > 0.

deduction\_gap(S, Gap) :-

\+ member(S, ['80C', 'EPF', 'LifeInsurance', 'NPS', '80D']),

(deduction(S, Used) -> true ; Used = 0),

max\_deduction(S, Max),

Gap is Max - Used,

Gap > 0.

% -----------------------------

% Deduction Tips

print\_deduction\_tips :-

deduction\_gap(S, G),

format("💡 Tip: Invest ₹~w more in ~w to save tax.~n", [G, S]),

fail.

print\_deduction\_tips.

% -----------------------------

% Regime Explanation

explain\_choice(OldTax, NewTax, Deductions) :-

Diff is OldTax - NewTax,

Diff > 0,

format("📌 The New Regime is suggested as it saves ₹~2f more tax than the Old Regime. You claimed deductions of ₹~w.~n", [Diff, Deductions]).

explain\_choice(OldTax, NewTax, Deductions) :-

Diff is NewTax - OldTax,

Diff >= 0,

format("📌 The Old Regime is suggested as it saves ₹~2f more tax due to your total deductions of ₹~w.~n", [Diff, Deductions]).

% -----------------------------

% Final Tax Summary

tax\_summary :-

unify\_example,

income(Income),

age(Age),

total\_deductions(TotalDeduction),

compute\_tax(old, OldTax),

compute\_tax(new, NewTax),

% Store computed tax values for Python

retractall(old\_regime\_tax(\_)),

retractall(new\_regime\_tax(\_)),

assertz(old\_regime\_tax(OldTax)),

assertz(new\_regime\_tax(NewTax)),

suggest\_regime(BestRegime, OldTax, NewTax),

taxable\_income(BestRegime, TaxableIncome),

format("Tax Analysis Suggestion:~n~n"),

format("Income: ₹~w~n", [Income]),

format("Age: ~w~n", [Age]),

format("Deductions (incl. ₹50,000 standard): ₹~w~n", [TotalDeduction]),

format("Taxable Income: ₹~w~n", [TaxableIncome]),

format("Old Regime Tax: ₹~2f~n", [OldTax]),

format("New Regime Tax: ₹~2f~n", [NewTax]),

format("Suggested Regime: ~w~n", [BestRegime]),

explain\_choice(OldTax, NewTax, TotalDeduction),

print\_deduction\_tips,

show\_text\_chart(OldTax, NewTax).

**PYTHON CODE (app.py):**

# app.py

from flask import Flask, render\_template, request

from pyswip import Prolog

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import io, base64, threading

app = Flask(\_\_name\_\_)

app.secret\_key = "replace-me-with-a-secret"

# Prolog initialization

PROLOG\_FILE = "tax\_advisor2.pl"

prolog = Prolog()

prolog.consult(PROLOG\_FILE)

# Simple lock to protect Prolog interactions (pyswip & SWI-Prolog are not fully thread-safe)

prolog\_lock = threading.Lock()

def run\_prolog\_clear\_and\_assert(age\_val, income\_val, deductions\_dict):

    """

    Clear previous user facts and assert new ones for age, income and deductions.

    """

    with prolog\_lock:

        # clear previous facts

        list(prolog.query("retractall(income(\_))"))

        list(prolog.query("retractall(age(\_))"))

        list(prolog.query("retractall(deduction(\_, \_))"))

        list(prolog.query("retractall(old\_regime\_tax(\_))"))

        list(prolog.query("retractall(new\_regime\_tax(\_))"))

        # assert current facts

        prolog.assertz(f"age({age\_val})")

        prolog.assertz(f"income({income\_val})")

        for sec, amt in deductions\_dict.items():

            # ensure section name matches Prolog facts (quote it)

            prolog.assertz(f"deduction('{sec}', {amt})")

def call\_tax\_summary\_and\_collect():

    """

    Run tax\_summary/0 and collect values we need from Prolog.

    Returns a dict of values.

    """

    with prolog\_lock:

        # run the main prolog summary which also asserts old\_regime\_tax/1 and new\_regime\_tax/1

        list(prolog.query("tax\_summary."))

        # read old and new tax (dynamic predicates asserted by tax\_summary)

        old\_tax\_q = list(prolog.query("old\_regime\_tax(T)"))

        new\_tax\_q = list(prolog.query("new\_regime\_tax(T)"))

        old\_tax = float(old\_tax\_q[0]['T']) if old\_tax\_q else 0.0

        new\_tax = float(new\_tax\_q[0]['T']) if new\_tax\_q else 0.0

        # total deductions (Prolog predicate)

        td\_q = list(prolog.query("total\_deductions(TD)"))

        total\_deductions = int(td\_q[0]['TD']) if td\_q else None

        # taxable incomes (old and new)

        ti\_old\_q = list(prolog.query("taxable\_income(old, TIold)"))

        ti\_new\_q = list(prolog.query("taxable\_income(new, TInew)"))

        ti\_old = int(ti\_old\_q[0]['TIold']) if ti\_old\_q else None

        ti\_new = int(ti\_new\_q[0]['TInew']) if ti\_new\_q else None

        # suggested regime

        sr\_q = list(prolog.query("suggest\_regime(R, OT, NT)"))

        suggested = sr\_q[0]['R'] if sr\_q else None

        # deduction tips (multiple solutions)

        tips = []

        for sol in prolog.query("deduction\_gap(Sec, Gap)"):

            tips.append((sol['Sec'], int(sol['Gap'])))

    return {

        "old\_tax": old\_tax,

        "new\_tax": new\_tax,

        "total\_deductions": total\_deductions,

        "ti\_old": ti\_old,

        "ti\_new": ti\_new,

        "suggested\_regime": suggested,

        "tips": tips

    }

def make\_chart\_base64(old\_tax, new\_tax):

    """Return a PNG image as base64 string comparing old vs new taxes."""

    regimes = ["Old Regime", "New Regime"]

    taxes = [old\_tax, new\_tax]

    colors = ['#1976D2', '#FF8F00']  # blue / orange

    fig, ax = plt.subplots(figsize=(6, 4.2), dpi=120)

    bars = ax.bar(regimes, taxes, color=colors, edgecolor='black', linewidth=1)

    # highlight cheaper regime

    cheaper\_index = int(taxes.index(min(taxes)))

    bars[cheaper\_index].set\_edgecolor('gold')

    bars[cheaper\_index].set\_linewidth(3)

    # value labels

    for bar in bars:

        h = bar.get\_height()

        ax.text(bar.get\_x() + bar.get\_width() / 2, h + max(1, max(taxes)\*0.01),

                f"₹{h:,.0f}", ha='center', va='bottom', fontsize=10, fontweight='bold')

    # zooming to emphasize small differences:

    min\_val = min(taxes)

    max\_val = max(taxes)

    if max\_val > 0:

        margin = max((max\_val - min\_val) \* 0.25, max\_val \* 0.02)

        lower = max(min\_val - margin, 0)

        upper = max\_val + margin

        ax.set\_ylim(lower, upper)

    ax.set\_ylabel("Tax Payable (₹)")

    ax.set\_title("Old vs New Regime Tax Comparison")

    ax.grid(axis='y', linestyle='--', alpha=0.5)

    buf = io.BytesIO()

    plt.tight\_layout()

    fig.savefig(buf, format='png', bbox\_inches='tight')

    plt.close(fig)

    buf.seek(0)

    img\_b64 = base64.b64encode(buf.read()).decode('ascii')

    return img\_b64

# Route handlers

@app.route('/', methods=['GET'])

def index():

    return render\_template('index.html')

@app.route('/result', methods=['POST'])

def result():

    try:

        # read basic fields

        age = int(request.form.get('age', '0') or 0)

        income = int(request.form.get('income', '0') or 0)

        # Collect deduction fields from the form. Map form names -> Prolog section names

        deductions = {}

        # form field names from your index.html:

        # "80C", "EPF", "80D", "80CCD\_1B", "LIFE"

        form\_mapping = {

            "80C": "80C",

            "EPF": "EPF",

            "80D": "80D",

            "80CCD\_1B": "80CCD(1B)",

            "LIFE": "Life Insurance"

        }

        for form\_field, section\_name in form\_mapping.items():

            v = request.form.get(form\_field)

            if v is None or v == "":

                continue

            try:

                amt = int(v)

                if amt > 0:

                    deductions[section\_name] = amt

            except ValueError:

                # ignore invalid numbers, or you can flash an error

                pass

        # Tell Prolog the facts

        run\_prolog\_clear\_and\_assert(age, income, deductions)

        # Run tax logic and collect outputs

        prolog\_results = call\_tax\_summary\_and\_collect()

        old\_tax = prolog\_results['old\_tax']

        new\_tax = prolog\_results['new\_tax']

        total\_deductions = prolog\_results['total\_deductions']

        ti\_old = prolog\_results['ti\_old']

        ti\_new = prolog\_results['ti\_new']

        suggested = prolog\_results['suggested\_regime']

        tips = prolog\_results['tips']

        # make chart as base64 PNG

        chart\_png = make\_chart\_base64(old\_tax, new\_tax)

        # prepare data for template

        return render\_template(

            'result.html',

            income=income,

            age=age,

            deductions=deductions,

            total\_deductions=total\_deductions,

            ti\_old=ti\_old,

            ti\_new=ti\_new,

            old\_tax=old\_tax,

            new\_tax=new\_tax,

            suggested\_regime=suggested,

            tips=tips,

            chart\_png=chart\_png

        )

    except Exception as e:

        # On error, show index with message

        return render\_template('index.html', error=str(e))

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

**HTML CODE:**

**index.html:**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Tax Advisor</title>

  <meta name="viewport" content="width=device-width,initial-scale=1">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

  <style>

    body {

      background: linear-gradient(120deg, #f6f9fc, #e9f2f9);

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    }

    .card {

      border-radius: 15px;

      box-shadow: 0 4px 15px rgba(0,0,0,0.1);

    }

    .btn-custom {

      background-color: #007bff;

      color: white;

      font-weight: bold;

    }

    .btn-custom:hover {

      background-color: #0056b3;

    }

    .hint {

      font-size: 0.85em;

      color: #6c757d;

    }

    .invalid-input {

      border-color: #dc3545 !important; /\* Bootstrap danger red \*/

      background-color: #f8d7da !important;

    }

    .warning-text {

      color: #dc3545;

      font-weight: 600;

      font-size: 0.9em;

    }

  </style>

</head>

<body>

  <div class="container py-5">

    <div class="row justify-content-center">

      <div class="col-lg-8 col-md-10">

        <div class="card p-4">

          <h2 class="text-center mb-3">💰 Tax Bracket Advisor</h2>

          <p class="text-center text-muted">

            Fill in your details and available deductions to see which tax regime is better for you.

          </p>

          <form id="taxForm" method="post" action="{{ url\_for('result') }}" novalidate>

            <div class="mb-3">

              <label class="form-label">Age</label>

              <input required name="age" id="age" type="number" min="18" max="120" class="form-control" placeholder="e.g. 30">

              <div id="ageError" class="warning-text d-none">Age must be between 18 and 120.</div>

            </div>

            <div class="mb-3">

              <label class="form-label">Annual Income (₹)</label>

              <input required name="income" id="income" type="number" min="0" class="form-control" placeholder="e.g. 1200000">

              <div id="incomeError" class="warning-text d-none">Income must be a positive number.</div>

            </div>

            <h5 class="mt-4">Deductions (optional)</h5>

            <div class="row">

              <div class="col-md-6 mb-3">

                <label class="form-label">80C (LIC / PPF / EPF) – ₹150,000 limit</label>

                <input name="80C" id="ded80C" type="number" min="0" max="150000" class="form-control" placeholder="e.g. 120000">

                <div id="ded80CError" class="warning-text d-none">Exceeds limit of ₹150,000.</div>

              </div>

              <div class="col-md-6 mb-3">

                <label class="form-label">EPF (part of 80C)</label>

                <input name="EPF" id="dedEPF" type="number" min="0" class="form-control" placeholder="optional">

              </div>

            </div>

            <div class="row">

              <div class="col-md-6 mb-3">

                <label class="form-label">80D (Medical Insurance)</label>

                <input name="80D" id="ded80D" type="number" min="0" class="form-control" placeholder="e.g. 25000">

                <div class="hint">Limit: ₹25,000 (<60 yrs) or ₹50,000 (≥60 yrs)</div>

                <div id="ded80DError" class="warning-text d-none">Exceeds allowed limit for your age.</div>

              </div>

              <div class="col-md-6 mb-3">

                <label class="form-label">80CCD(1B) / NPS</label>

                <input name="80CCD\_1B" id="dedNPS" type="number" min="0" class="form-control" placeholder="e.g. 50000">

              </div>

            </div>

            <div class="mb-3">

              <label class="form-label">Life Insurance (part of 80C)</label>

              <input name="LIFE" id="dedLife" type="number" min="0" class="form-control" placeholder="optional">

            </div>

            <div class="d-grid">

              <button type="submit" class="btn btn-custom btn-lg">Calculate & Show Recommendation</button>

            </div>

          </form>

        </div>

      </div>

    </div>

  </div>

<script>

  (() => {

    const form = document.getElementById('taxForm');

    // Elements

    const ageInput = document.getElementById('age');

    const incomeInput = document.getElementById('income');

    const ded80CInput = document.getElementById('ded80C');

    const dedEPFInput = document.getElementById('dedEPF');

    const ded80DInput = document.getElementById('ded80D');

    const dedNPSInput = document.getElementById('dedNPS');

    const dedLifeInput = document.getElementById('dedLife');

    // Error elements

    const ageError = document.getElementById('ageError');

    const incomeError = document.getElementById('incomeError');

    const ded80CError = document.getElementById('ded80CError');

    const ded80DError = document.getElementById('ded80DError');

    // Utility functions

    function setError(elem, errorElem, condition) {

      if (condition) {

        elem.classList.add('invalid-input');

        errorElem.classList.remove('d-none');

      } else {

        elem.classList.remove('invalid-input');

        errorElem.classList.add('d-none');

      }

    }

    // Validation

    function validateAge() {

      const val = parseInt(ageInput.value);

      setError(ageInput, ageError, isNaN(val) || val < 18 || val > 120);

      return !isNaN(val) && val >= 18 && val <= 120;

    }

    function validateIncome() {

      const val = parseInt(incomeInput.value);

      setError(incomeInput, incomeError, isNaN(val) || val < 0);

      return !isNaN(val) && val >= 0;

    }

    function validateDeductions() {

      let valid = true;

      // 80C max 150,000

      const val80C = parseInt(ded80CInput.value) || 0;

      setError(ded80CInput, ded80CError, val80C > 150000);

      if (val80C > 150000) valid = false;

      // 80D limit depends on age: 25,000 or 50,000

      const ageVal = parseInt(ageInput.value);

      const val80D = parseInt(ded80DInput.value) || 0;

      if (!isNaN(ageVal)) {

        const max80D = ageVal >= 60 ? 50000 : 25000;

        setError(ded80DInput, ded80DError, val80D > max80D);

        if (val80D > max80D) valid = false;

      } else {

        // If age not valid, clear 80D error

        setError(ded80DInput, ded80DError, false);

      }

      return valid;

    }

    // Attach events

    ageInput.addEventListener('input', () => {

      validateAge();

      validateDeductions();

    });

    incomeInput.addEventListener('input', validateIncome);

    ded80CInput.addEventListener('input', validateDeductions);

    ded80DInput.addEventListener('input', validateDeductions);

    form.addEventListener('submit', e => {

      let valid = validateAge() && validateIncome() && validateDeductions();

      if (!valid) {

        e.preventDefault();

        alert("Please fix the highlighted errors before submitting.");

      }

    });

  })();

</script>

</body>

</html>

**Result.html:**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Tax Recommendation</title>

  <meta name="viewport" content="width=device-width,initial-scale=1">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

  <style>

    body {

      background: linear-gradient(120deg, #f6f9fc, #e9f2f9);

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    }

    .card {

      border-radius: 15px;

      box-shadow: 0 4px 15px rgba(0,0,0,0.1);

    }

    .tip {

      background: #fff8e1;

      border-left: 4px solid #ffb300;

      padding: 10px;

      border-radius: 5px;

      margin-bottom: 8px;

      font-size: 0.95em;

    }

    .chart img {

      max-width: 100%;

      border-radius: 8px;

    }

    /\* Colored regime suggestion card \*/

    .regime-card {

      border-radius: 12px;

      padding: 20px;

      color: white;

      font-weight: 600;

      font-size: 1.3rem;

      text-align: center;

      margin-bottom: 24px;

      box-shadow: 0 4px 12px rgba(0,0,0,0.1);

    }

    .regime-old {

      background: #198754; /\* Bootstrap success green \*/

      border: 2px solid #145c32;

    }

    .regime-new {

      background: #0d6efd; /\* Bootstrap primary blue \*/

      border: 2px solid #084298;

    }

  </style>

</head>

<body>

  <div class="container py-5">

    <div class="row justify-content-center">

      <div class="col-lg-10 col-md-12">

        <div class="card p-4">

          <h2 class="text-center mb-4">📊 Tax Recommendation</h2>

          <!-- Regime suggestion colored card -->

          {% if suggested\_regime == 'old' %}

            <div class="regime-card regime-old">

              Recommended Regime: Old Regime ✅

            </div>

          {% elif suggested\_regime == 'new' %}

            <div class="regime-card regime-new">

              Recommended Regime: New Regime ✅

            </div>

          {% endif %}

          <h4 class="mb-3">Summary</h4>

          <table class="table table-striped">

            <tr><th>Income</th><td>₹{{ income|int }}</td></tr>

            <tr><th>Age</th><td>{{ age }}</td></tr>

            <tr>

              <th>Claimed Deductions</th>

              <td>

                {% if deductions %}

                  <ul class="mb-0">

                  {% for k,v in deductions.items() %}

                    <li>{{ k }}: ₹{{ v }}</li>

                  {% endfor %}

                  </ul>

                {% else %}

                  ₹0

                {% endif %}

              </td>

            </tr>

            <tr><th>Total Deductions (incl. ₹50,000 standard)</th><td>₹{{ total\_deductions }}</td></tr>

            <tr><th>Taxable Income (old)</th><td>₹{{ ti\_old }}</td></tr>

            <tr><th>Taxable Income (new)</th><td>₹{{ ti\_new }}</td></tr>

            <tr><th>Old Regime Tax</th><td>₹{{ old\_tax|float | round(2) }}</td></tr>

            <tr><th>New Regime Tax</th><td>₹{{ new\_tax|float | round(2) }}</td></tr>

            <tr><th>Suggested Regime</th><td><strong>{{ suggested\_regime }}</strong></td></tr>

          </table>

          <h4 class="mt-4">💡 Tax-saving Tips</h4>

          {% if tips %}

            {% for s,g in tips %}

              <div class="tip">Invest ₹{{ g }} more in <strong>{{ s }}</strong> to save tax.</div>

            {% endfor %}

          {% else %}

            <div class="alert alert-success">No direct tips — you’re already using your deductions well.</div>

          {% endif %}

          <div class="chart text-center mt-4">

            {% if chart\_png %}

              <img src="data:image/png;base64,{{ chart\_png }}" alt="Tax chart">

            {% else %}

              <div class="alert alert-warning">Chart not available.</div>

            {% endif %}

          </div>

          <div class="text-center mt-4">

            <a href="{{ url\_for('index') }}" class="btn btn-primary btn-lg">🔄 Run Again</a>

          </div>

        </div>

      </div>

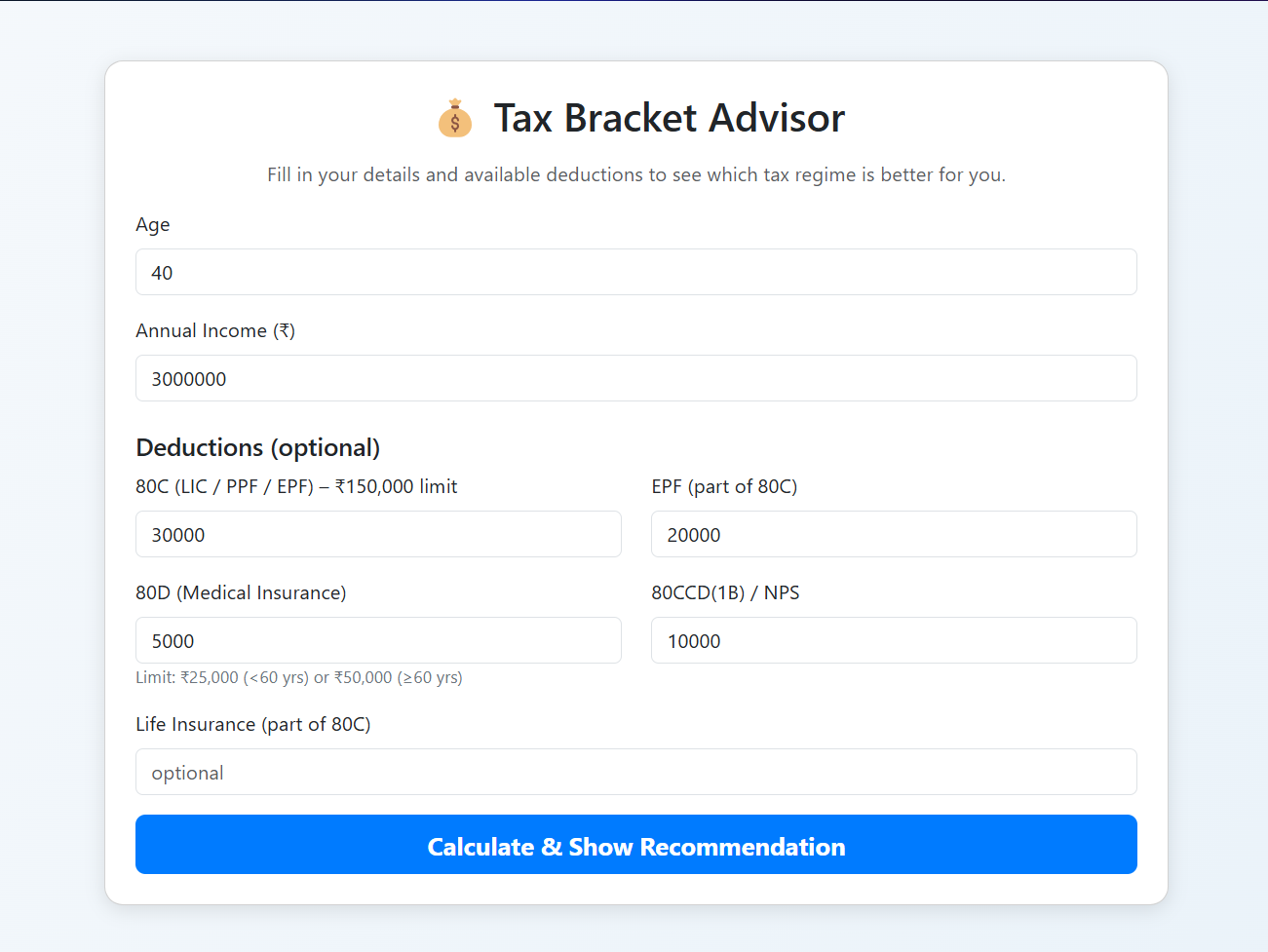
    </div>

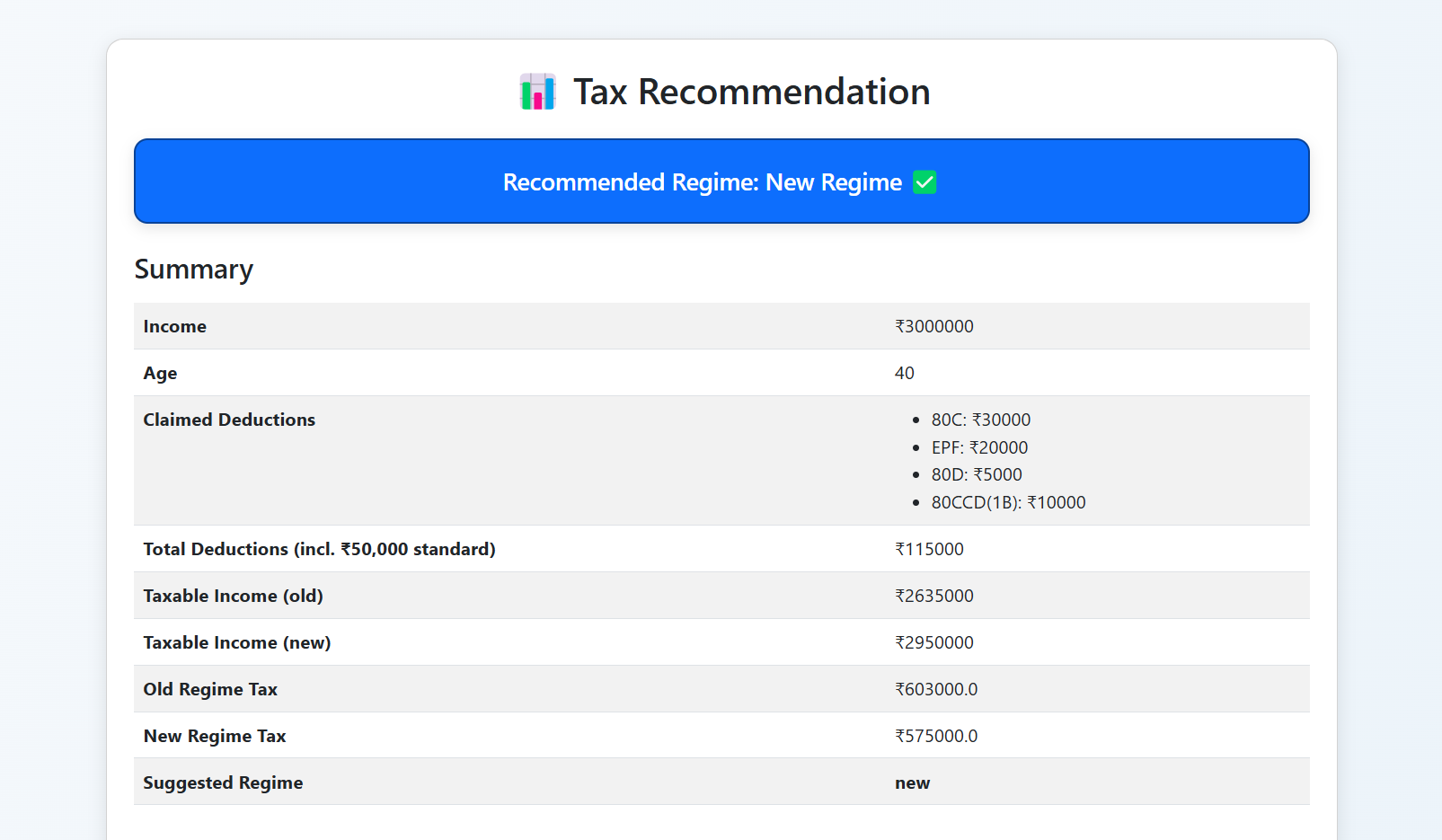
  </div>

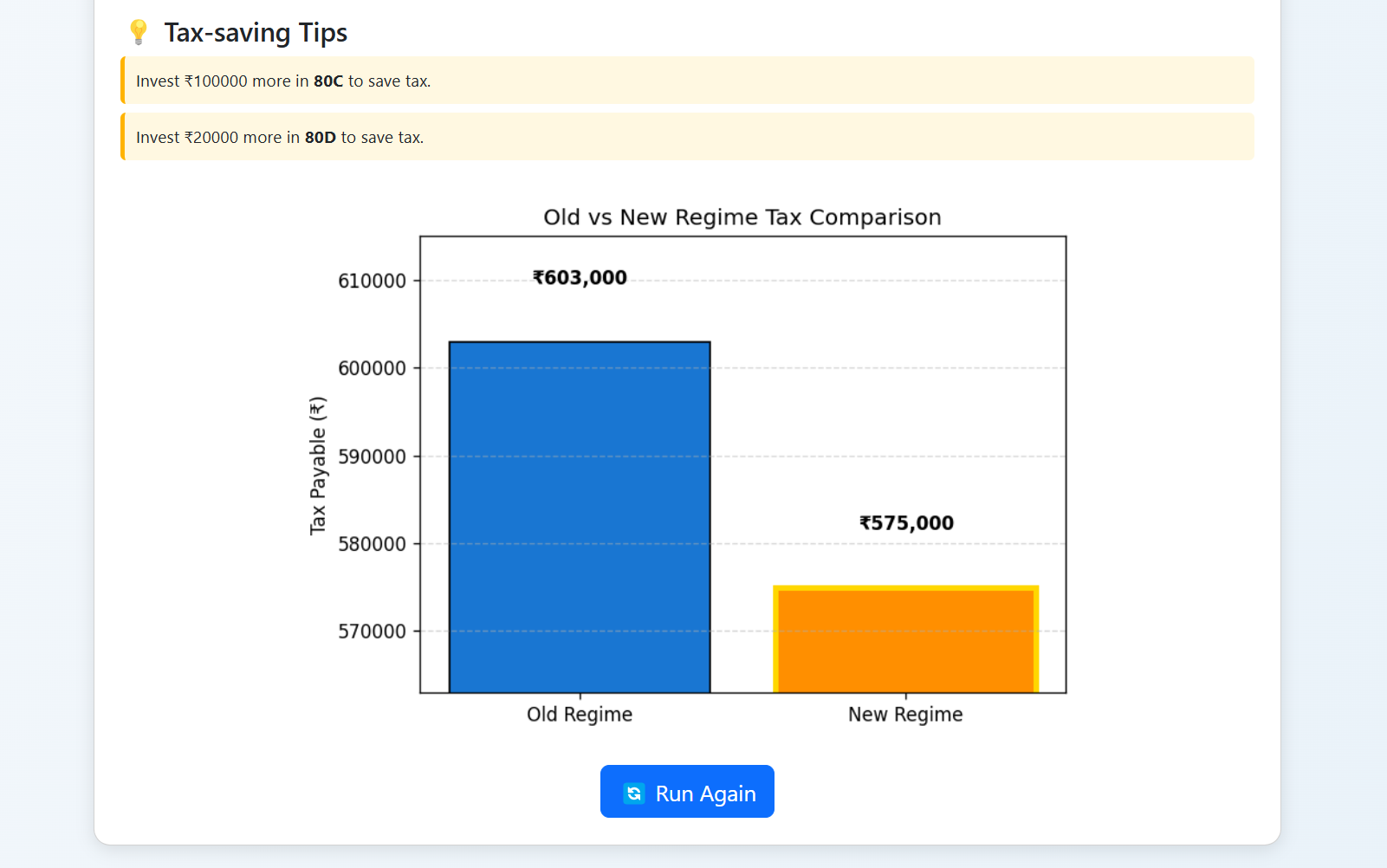
</body>

</html>

**OUTPUT:**

****

****

****