Base.html

<!-- templates/base.html -->

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Billing Dashboard</title>

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

  <!-- Bootstrap CSS -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

  <!-- Tom Select -->

  <link href="https://cdn.jsdelivr.net/npm/tom-select@2.3.1/dist/css/tom-select.css" rel="stylesheet">

  <style>

    body {

      background-color: #f8f9fa;

      font-family: "Segoe UI", Roboto, sans-serif;

    }

    .navbar {

      box-shadow: 0 2px 6px rgba(0,0,0,0.1);

    }

    .navbar-brand img {

      height: 40px; /\* Adjusted logo size \*/

    }

    .nav-link.active {

      font-weight: 600;

      color: #0d6efd !important;

    }

    .container {

      margin-top: 2rem;

    }

    .alert {

      border-radius: 0.75rem;

      box-shadow: 0 2px 8px rgba(0,0,0,0.05);

    }

    footer {

      margin-top: 3rem;

      padding: 1rem 0;

      text-align: center;

      font-size: 0.9rem;

      color: #6c757d;

      border-top: 1px solid #dee2e6;

    }

  </style>

</head>

<body>

  <!-- Navbar -->

  <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">

    <div class="container-fluid">

      <a class="navbar-brand d-flex align-items-center" href="{{ url\_for('home') }}">

        <img src="{{ url\_for('static', filename='505616.png') }}" alt="Logo" class="me-2">

        <span class="fw-bold">Billing Dashboard</span>

      </a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"

              data-bs-target="#navbarNav" aria-controls="navbarNav"

              aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav ms-auto">

          <li class="nav-item">

            <a class="nav-link{% if request.endpoint == 'dashboard' %} active{% endif %}"

               href="{{ url\_for('dashboard') }}">Dashboard</a>

          </li>

          <li class="nav-item">

            <a class="nav-link{% if request.endpoint == 'list\_clients' %} active{% endif %}"

               href="{{ url\_for('list\_clients') }}">Clients</a>

          </li>

          <li class="nav-item">

            <a class="nav-link{% if request.endpoint == 'bills' %} active{% endif %}"

               href="{{ url\_for('bills') }}">Bills</a>

          </li>

          <li class="nav-item">

            <a class="nav-link{% if request.endpoint == 'receipts' %} active{% endif %}"

               href="{{ url\_for('receipts') }}">Receipts</a>

          </li>

        </ul>

      </div>

    </div>

  </nav>

  <!-- Flash Messages -->

  <div class="container">

    {% with messages = get\_flashed\_messages(with\_categories=true) %}

      {% if messages %}

        {% for category, message in messages %}

          <div class="alert alert-{{ category }} alert-dismissible fade show mt-3" role="alert">

            {{ message }}

            <button type="button" class="btn-close" data-bs-dismiss="alert"></button>

          </div>

        {% endfor %}

      {% endif %}

    {% endwith %}

    <!-- Main Page Content -->

    {% block content %}{% endblock %}

  </div>

  <!-- Footer -->

  <footer>

    &copy; {{ current\_year if current\_year else "2025" }} Billing Dashboard · All rights reserved

  </footer>

  <!-- Bootstrap JS -->

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

  <!-- Tom Select JS -->

  <script src="https://cdn.jsdelivr.net/npm/tom-select@2.3.1/dist/js/tom-select.complete.min.js"></script>

  <!-- Popover Initialization -->

  <script>

    document.addEventListener('DOMContentLoaded', function () {

      document.querySelectorAll('[data-bs-toggle="popover"]').forEach(function (el) {

        new bootstrap.Popover(el, {

          container: 'body',

          html: false,

          trigger: 'focus'

        });

      });

    });

  </script>

</body>

</html>

Bill\_edit.html

<!-- templates/bill\_edit.html -->

{% extends "base.html" %}

{% block content %}

<h3>Edit Bill</h3>

<form method="post" class="row g-2">

  <div class="col-md-3">

    <label class="form-label">Bill No</label>

    <input name="bill\_no" class="form-control" value="{{ b.bill\_no }}" required>

  </div>

  <div class="col-md-3">

    <label class="form-label">Bill Date</label>

    <input type="date" name="bill\_date" class="form-control" value="{{ b.bill\_date }}" required>

  </div>

  <div class="col-md-3">

  <label class="form-label" for="editBillClientCombobox">Client</label>

  <!-- Text input behaves as a combobox -->

  <input

    id="editBillClientCombobox"

    class="form-control mb-1"

    type="text"

    placeholder="Type to find client"

    role="combobox"

    aria-autocomplete="list"

    aria-haspopup="listbox"

    aria-expanded="false"

    aria-controls="editBillClientListbox"

    aria-activedescendant=""

    autocomplete="off">

  <!-- List of clients as a listbox -->

  <div id="editBillClientListbox"

       role="listbox"

       class="list-group"

       style="max-height:180px; overflow:auto;"

       hidden>

    {% for c in clients %}

      <button type="button"

              class="list-group-item list-group-item-action"

              role="option"

              id="edit-bill-client-opt-{{ c.id }}"

              data-id="{{ c.id }}"

              aria-selected="{{ 'true' if c.id==b.client\_id else 'false' }}">

        {{ c.name }}

      </button>

    {% endfor %}

  </div>

  <!-- Hidden field submitted to server (pre-filled with current selection) -->

  <input type="hidden" name="client\_id" id="editBillClientHidden" value="{{ b.client\_id }}" required>

</div>

  <div class="col-md-3">

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

    <input type="number" step="0.01" name="amount" class="form-control" value="{{ b.amount }}" required>

  </div>

  <div class="col-12">

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

    <input name="description" class="form-control" value="{{ b.description or '' }}">

  </div>

  <div class="col-12">

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

    <input name="remarks" class="form-control" value="{{ b.remarks or '' }}">

  </div>

  <div class="col-12">

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

    <input name="Subject" class="form-control" value="{{ b.Subject or '' }}">

  </div>

  <div class="col-12">

    <button class="btn btn-primary">Update</button>

    <a class="btn btn-secondary ms-2" href="{{ url\_for('bills') }}">Cancel</a>

  </div>

</form>

<style>

  #editBillClientListbox .list-group-item:hover { background:#f2f6ff; }

</style>

<script>

(function(){

  const input = document.getElementById('editBillClientCombobox');

  const list  = document.getElementById('editBillClientListbox');

  const hid   = document.getElementById('editBillClientHidden');

  if (!input || !list || !hid) return;

  // Initialize input with the current client's name (from the selected option)

  // Server renders selected item; we read it and populate the text box on load.

  const preSelected = list.querySelector('[aria-selected="true"]');

  if (preSelected) {

    input.value = preSelected.textContent.trim();

  }

  let activeIndex = -1;

  let visible = [];

  const optsAll = () => Array.from(list.querySelectorAll('[role="option"]'));

  function openList() {

    list.hidden = false;

    input.setAttribute('aria-expanded','true');

    rebuildVisible();

  }

  function closeList() {

    list.hidden = true;

    input.setAttribute('aria-expanded','false');

    setActive(-1);

  }

  function setActive(i) {

    visible.forEach(o=>o.classList.remove('active'));

    activeIndex = i;

    if (i>=0 && i<visible.length) {

      const el = visible[i];

      el.classList.add('active');

      input.setAttribute('aria-activedescendant', el.id);

      const r = el.getBoundingClientRect();

      const lr = list.getBoundingClientRect();

      if (r.top < lr.top) list.scrollTop -= (lr.top - r.top);

      if (r.bottom > lr.bottom) list.scrollTop += (r.bottom - lr.bottom);

    } else {

      input.removeAttribute('aria-activedescendant');

    }

  }

  function rebuildVisible() {

    const q = (input.value || '').toLowerCase();

    visible = [];

    optsAll().forEach(btn=>{

      const show = btn.textContent.toLowerCase().includes(q);

      btn.style.display = show ? '' : 'none';

      btn.setAttribute('aria-selected','false');

      if (show) visible.push(btn);

    });

    setActive(visible.length ? 0 : -1);

  }

  function commit(btn) {

    if (!btn) return;

    hid.value = btn.dataset.id;

    input.value = btn.textContent.trim();

    visible.forEach(o=>o.setAttribute('aria-selected', o===btn ? 'true' : 'false'));

    closeList();

  }

  // Typing opens and filters

  input.addEventListener('input', ()=>{ openList(); rebuildVisible(); });

  input.addEventListener('focus', openList);

  input.addEventListener('blur', ()=> setTimeout(()=>{ if (!list.contains(document.activeElement)) closeList(); }, 120));

  // Keyboard navigation

  input.addEventListener('keydown', (e)=>{

    if (e.key==='ArrowDown') { e.preventDefault(); if (list.hidden) openList(); if (visible.length) setActive(Math.min(activeIndex+1, visible.length-1)); }

    else if (e.key==='ArrowUp') { e.preventDefault(); if (visible.length) setActive(Math.max(activeIndex-1, 0)); }

    else if (e.key==='Enter') { if (!list.hidden && activeIndex>=0) { e.preventDefault(); commit(visible[activeIndex]); } }

    else if (e.key==='Escape') { closeList(); }

  });

  // Mouse selection

  list.addEventListener('mousedown', e=>e.preventDefault());

  list.addEventListener('click', e=>{

    const btn = e.target.closest('[role="option"]');

    if (btn) commit(btn);

  });

})();

</script>

{% endblock %}

bills.html

{% extends "base.html" %}

{% block content %}

<h3>Add Bill</h3>

<form method="post" class="row g-2" id="billForm">

  <div class="col-md-3">

    <label class="form-label" for="billClientCombobox">Client</label>

    <input

      id="billClientCombobox"

      class="form-control mb-1"

      type="text"

      placeholder="Type to find client"

      role="combobox"

      aria-autocomplete="list"

      aria-haspopup="listbox"

      aria-expanded="false"

      aria-controls="billClientListbox"

      aria-activedescendant=""

      autocomplete="off">

    <div id="billClientListbox"

         role="listbox"

         class="list-group"

         style="max-height:180px; overflow:auto;"

         hidden>

      {% for c in clients %}

        <button type="button"

                class="list-group-item list-group-item-action"

                role="option"

                id="bill-client-opt-{{ c.id }}"

                data-id="{{ c.id }}"

                aria-selected="false">

          {{ c.name }}

        </button>

      {% endfor %}

    </div>

    <input type="hidden" name="client\_id" id="billClientHidden" required>

  </div>

  <div class="col-md-3">

    <label class="form-label">Bill No</label>

    <input name="bill\_no" class="form-control" required>

  </div>

  <div class="col-md-3">

    <label class="form-label">Bill Date</label>

    <input type="date" name="bill\_date" class="form-control" required>

  </div>

  <div class="col-md-3">

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

    <input type="number" step="0.01" name="amount" class="form-control" required>

  </div>

  <div class="col-12">

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

    <input name="Subject" class="form-control">

  </div>

  <div class="col-12">

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

    <input name="description" class="form-control">

  </div>

  <div class="col-12">

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

    <input name="remarks" class="form-control">

  </div>

  <div class="col-12">

    <button class="btn btn-primary">Save Bill</button>

  </div>

</form>

<hr>

<h3>All Bills</h3>

<!-- Replace the two separate inputs with a single combined search input (like recSearch) -->

<input id="billSearch" class="form-control mb-2" placeholder="Search by client or bill no">

<table class="table table-sm table-striped">

  <thead>

    <tr>

      <th>Date</th><th>Bill No</th><th>Client</th><th>Amount</th>

      <th>Description</th><th>Remarks</th><th>Subject</th>

    </tr>

  </thead>

  <tbody id="billTable">

    {% for b in bills %}

    <tr>

      <td>{{ b.bill\_date }}</td>

      <td>{{ b.bill\_no }}</td>

      <td>{{ b.client.name }}</td>

      <td>{{ "%.2f"|format(b.amount) }}</td>

      <td>

        {% set desc = b.description or '' %}

        {% if desc|length > 25 %}

          {{ desc[:25] }}…

          <button type="button" class="btn btn-sm btn-link p-0"

                  data-bs-toggle="popover" data-bs-trigger="focus"

                  title="Description" data-bs-content="{{ desc|e }}">view</button>

        {% else %}

          {{ desc }}

        {% endif %}

      </td>

      <td>

        {% set rem = b.remarks or '' %}

        {% if rem|length > 25 %}

          {{ rem[:25] }}…

          <button type="button" class="btn btn-sm btn-link p-0"

                  data-bs-toggle="popover" data-bs-trigger="focus"

                  title="Remarks" data-bs-content="{{ rem|e }}">view</button>

        {% else %}

          {{ rem }}

        {% endif %}

      </td>

      <td>

        <a class="btn btn-sm btn-outline-primary me-1" href="{{ url\_for('edit\_bill', bid=b.id) }}">Edit</a>

        <form method="post" action="{{ url\_for('delete\_row', table='bill', row\_id=b.id) }}" class="d-inline" onsubmit="return confirm('Delete bill?')">

          <button class="btn btn-sm btn-outline-danger">Delete</button>

        </form>

      </td>

    </tr>

    {% endfor %}

  </tbody>

</table>

<style>

  #billClientListbox .list-group-item:hover { background:#f2f6ff; }

</style>

<script>

// -------- Client combobox for Add Bill (accessible, keyboard + mouse) ----------

(function(){

  const input = document.getElementById('billClientCombobox');

  const list  = document.getElementById('billClientListbox');

  const hid   = document.getElementById('billClientHidden');

  if (!input || !list || !hid) return;

  let activeIndex = -1;

  let visible = [];

  const optsAll = () => Array.from(list.querySelectorAll('[role="option"]'));

  function openList() {

    list.hidden = false;

    input.setAttribute('aria-expanded','true');

    rebuildVisible();

  }

  function closeList() {

    list.hidden = true;

    input.setAttribute('aria-expanded','false');

    setActive(-1);

  }

  function setActive(i) {

    visible.forEach(o=>o.classList.remove('active'));

    activeIndex = i;

    if (i>=0 && i<visible.length) {

      const el = visible[i];

      el.classList.add('active');

      input.setAttribute('aria-activedescendant', el.id);

      const r = el.getBoundingClientRect();

      const lr = list.getBoundingClientRect();

      if (r.top < lr.top) list.scrollTop -= (lr.top - r.top);

      if (r.bottom > lr.bottom) list.scrollTop += (r.bottom - lr.bottom);

    } else {

      input.removeAttribute('aria-activedescendant');

    }

  }

  function rebuildVisible() {

    const q = (input.value || '').toLowerCase();

    visible = [];

    optsAll().forEach(btn=>{

      const show = btn.textContent.toLowerCase().includes(q);

      btn.style.display = show ? '' : 'none';

      btn.setAttribute('aria-selected','false');

      if (show) visible.push(btn);

    });

    setActive(visible.length ? 0 : -1);

  }

  function commit(btn) {

    if (!btn) return;

    hid.value = btn.dataset.id;

    input.value = btn.textContent.trim();

    visible.forEach(o=>o.setAttribute('aria-selected', o===btn ? 'true' : 'false'));

    closeList();

  }

  input.addEventListener('input', ()=>{ openList(); rebuildVisible(); });

  input.addEventListener('focus', openList);

  input.addEventListener('blur', ()=> setTimeout(()=>{ if (!list.contains(document.activeElement)) closeList(); }, 120));

  input.addEventListener('keydown', (e)=>{

    if (e.key==='ArrowDown') { e.preventDefault(); if (list.hidden) openList(); if (visible.length) setActive(Math.min(activeIndex+1, visible.length-1)); }

    else if (e.key==='ArrowUp') { e.preventDefault(); if (visible.length) setActive(Math.max(activeIndex-1, 0)); }

    else if (e.key==='Enter') { if (!list.hidden && activeIndex>=0) { e.preventDefault(); commit(visible[activeIndex]); } }

    else if (e.key==='Escape') { closeList(); }

  });

  list.addEventListener('mousedown', e=>e.preventDefault());

  list.addEventListener('click', e=>{

    const btn = e.target.closest('[role="option"]');

    if (btn) commit(btn);

  });

})();

// -------- Table-based search (Client + Bill No) ----------

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

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

function filterBills() {

  const qc = (billClientSearch.value || '').toLowerCase();

  const qb = (billNumberSearch.value || '').toLowerCase();

  document.querySelectorAll('#billTable tr').forEach(tr => {

    // columns: 0 Date, 1 Bill No, 2 Client, ...

    const billno = (tr.children[1]?.innerText || '').toLowerCase();

    const client = (tr.children[2]?.innerText || '').toLowerCase();

    const show = client.includes(qc) && billno.includes(qb);

    tr.style.display = show ? '' : 'none';

  });

}

billClientSearch.addEventListener('input', filterBills);

billNumberSearch.addEventListener('input', filterBills);

// -------- Enable Bootstrap popovers ----------

document.querySelectorAll('[data-bs-toggle="popover"]').forEach(el=>{

  new bootstrap.Popover(el, {container:'body', html:false});

});

</script>

<!-- Add this script (place it after the All Bills table) to wire up the single search box -->

<script>

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

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

    const q = (billSearch.value || '').toLowerCase();

    document.querySelectorAll('#billTable tr').forEach(tr => {

      // columns: 0 Date, 1 Bill No, 2 Client, ...

      const billNo  = (tr.children[1]?.innerText || '').toLowerCase();

      const client  = (tr.children[2]?.innerText || '').toLowerCase();

      const rowText = tr.innerText.toLowerCase();

      tr.style.display = (billNo.includes(q) || client.includes(q) || rowText.includes(q)) ? '' : 'none';

    });

  });

</script>

{% endblock %}

Client\_edit.html

{% extends "base.html" %}

{% block content %}

<h3>Edit Client</h3>

<form method="post" class="row g-2 mb-3">

  <div class="col-md-3">

    <input name="name" class="form-control" placeholder="Client name" value="{{ client.name }}" required>

  </div>

  <div class="col-md-3">

    <input name="address" class="form-control" placeholder="Address" value="{{ client.address or '' }}">

  </div>

  <div class="col-md-2">

    <input name="gst\_no" class="form-control" placeholder="GST No." value="{{ client.gst\_no or '' }}">

  </div>

  <div class="col-md-2">

    <input name="pan\_no" class="form-control" placeholder="PAN No." value="{{ client.pan\_no or '' }}">

  </div>

  <div class="col-md-2">

    <input name="remarks" class="form-control" placeholder="Remarks" value="{{ client.remarks or '' }}">

  </div>

  <div class="col-md-2">

    <button type="submit" class="btn btn-primary">Save</button>

  </div>

</form>

<a href="{{ url\_for('list\_clients') }}" class="btn btn-secondary">Cancel</a>

{% endblock %}

clients.html

{% extends "base.html" %}

{% block content %}

<h3>{% if client %}Edit Client{% else %}Add Client{% endif %}</h3>

<form method="post" action="{{ url\_for('add\_client' if not client else 'edit\_client', cid=client.id if client else None) }}" class="row g-2 mb-3">

  <div class="col-md-3">

    <input name="name" class="form-control" placeholder="Client name" value="{{ client.name if client else '' }}" required>

  </div>

  <div class="col-md-3">

    <input name="address" class="form-control" placeholder="Address" value="{{ client.address if client else '' }}">

  </div>

  <div class="col-md-2">

    <input name="gst\_no" class="form-control" placeholder="GST No." value="{{ client.gst\_no if client else '' }}">

  </div>

  <div class="col-md-2">

    <input name="pan\_no" class="form-control" placeholder="PAN No." value="{{ client.pan\_no if client else '' }}">

  </div>

  <div class="col-md-2">

    <input name="remarks" class="form-control" placeholder="Remarks" value="{{ client.remarks if client else '' }}">

  </div>

  <div class="col-12">

    <button type="submit" class="btn btn-primary">{% if client %}Update Client{% else %}Save Client{% endif %}</button>

  </div>

  {% if client %}

  <input type="hidden" name="cid" value="{{ client.id }}">

  {% endif %}

</form>

<hr>

<h3>All Clients</h3>

<input id="clientSearch" class="form-control mb-2" placeholder="Search by any field">

<table class="table table-sm table-striped">

  <thead>

    <tr>

      <th>Name</th>

      <th>Address</th>

      <th>GST No.</th>

      <th>PAN No.</th>

      <th>Remarks</th>

      <th style="width:150px;"></th>

    </tr>

  </thead>

  <tbody id="clientTable">

    {% for c in clients %}

    <tr>

      <td>{{ c.name }}</td>

      <td>

      {% set addr = c.address or '' %}

      {% if addr|length > 50 %}

        {{ addr[:50] }}…

        <button type="button" class="btn btn-sm btn-link p-0"

                data-bs-toggle="popover" data-bs-trigger="focus"

                title="Address" data-bs-content="{{ addr|e }}">view</button>

      {% else %}

        {{ addr }}

      {% endif %}

      </td>

      <td>

        {% set gst = c.gst\_no or '' %}

      {% if gst|length > 30 %}

        {{ gst[:30] }}…

        <button type="button" class="btn btn-sm btn-link p-0"

                data-bs-toggle="popover" data-bs-trigger="focus"

                title="GST No." data-bs-content="{{ gst|e }}">view</button>

      {% else %}

        {{ gst }}

      {% endif %}

      </td>

      <td>{{ c.pan\_no or '' }}</td>

      <td>

        {% set rmk = c.remarks or '' %}

        {% if rmk|length > 10 %}

          {{ rmk[:10] }}…

          <button type="button" class="btn btn-sm btn-link p-0"

                  data-bs-toggle="popover" data-bs-trigger="focus"

                  title="Remarks" data-bs-content="{{ rmk|e }}">view</button>

        {% else %}

          {{ rmk }}

        {% endif %}

      </td>

      <td>

        <a class="btn btn-sm btn-outline-primary me-1" href="{{ url\_for('edit\_client', cid=c.id) }}">Edit</a>

        <form method="post" action="{{ url\_for('delete\_row', table='client', row\_id=c.id) }}" class="d-inline" onsubmit="return confirm('Delete client?')">

          <button class="btn btn-sm btn-outline-danger">Delete</button>

        </form>

      </td>

    </tr>

    {% endfor %}

  </tbody>

</table>

<script>

// Search across all visible text in the row

const cs = document.getElementById('clientSearch');

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

  const q = (cs.value || '').toLowerCase();

  document.querySelectorAll('#clientTable tr').forEach(tr => {

    const text = tr.innerText.toLowerCase();

    tr.style.display = text.includes(q) ? '' : 'none';

  });

});

// Enable Bootstrap popovers for long remarks

document.querySelectorAll('[data-bs-toggle="popover"]').forEach(el => {

  new bootstrap.Popover(el, { container: 'body', html: false });

});

</script>

{% endblock %}

Dashboard.html

<!-- templates/dashboard.html — full file with client dropdown -->

{% extends "base.html" %}

{% block content %}

<h3>Dashboard</h3>

<form class="row g-2 mb-3" method="get">

  <div class="col-md-3">

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

    <select name="client" class="form-select">

      <option value="">All clients</option>

      {% for c in clients %}

      <option value="{{ c.name }}" {{ c.name==filters.client and 'selected' or '' }}>{{ c.name }}</option>

      {% endfor %}

    </select>

  </div>

  <div class="col-md-2">

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

    <input type="date" name="from" value="{{ filters.dfrom }}" class="form-control">

  </div>

  <div class="col-md-2">

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

    <input type="date" name="to" value="{{ filters.dto }}" class="form-control">

  </div>

  <div class="col-md-2">

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

    <select name="status" class="form-select">

      <option value="" {{ ''==filters.status and 'selected' or '' }}>All</option>

      <option value="Pending" {{ 'Pending'==filters.status and 'selected' or '' }}>Pending</option>

      <option value="Paid" {{ 'Paid'==filters.status and 'selected' or '' }}>Paid</option>

      <option value="Overpaid" {{ 'Overpaid'==filters.status and 'selected' or '' }}>Overpaid</option>

    </select>

  </div>

  <div class="col-md-3 d-flex align-items-end">

    <button class="btn btn-primary me-2">Apply</button>

    <a class="btn btn-outline-secondary" href="{{ url\_for('export\_reconciliation', fmt='csv') }}">Export CSV</a>

    <a class="btn btn-outline-secondary ms-2" href="{{ url\_for('export\_reconciliation', fmt='xlsx') }}">Export Excel</a>

  </div>

</form>

<div class="row mb-3">

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Total Bills</div>

      <div>₹ {{ "%.2f"|format(totals.total\_bills) }}</div>

    </div></div>

  </div>

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Total Paid</div>

      <div>₹ {{ "%.2f"|format(totals.total\_paid) }}</div>

    </div></div>

  </div>

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Total Balance</div>

      <div>₹ {{ "%.2f"|format(totals.total\_balance) }}</div>

    </div></div>

  </div>

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Pending</div>

      <div>{{ totals.count\_pending }}</div>

    </div></div>

  </div>

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Paid</div>

      <div>{{ totals.count\_paid }}</div>

    </div></div>

  </div>

  <div class="col-md-2">

    <div class="card"><div class="card-body">

      <div class="fw-bold">Overpaid</div>

      <div>{{ totals.count\_overpaid }}</div>

    </div></div>

  </div>

</div>

<table class="table table-sm table-striped">

  <thead>

    <tr>

      <th>Bill Date</th><th>Bill No</th><th>Client</th><th>Bill Amount</th><th>Paid</th><th>Balance</th><th>Status</th>

    </tr>

  </thead>

  <tbody>

    {% for r in rows %}

    <tr>

      <td>{{ r.bill\_date }}</td>

      <td>{{ r.bill\_no }}</td>

      <td>{{ r.client\_name }}</td>

      <td>{{ "%.2f"|format(r.amount) }}</td>

      <td>{{ "%.2f"|format(r.paid\_amount) }}</td>

      <td>{{ "%.2f"|format(r.balance) }}</td>

      <td>

        <span class="badge {% if r.status=='Paid' %}bg-success{% elif r.status=='Pending' %}bg-warning text-dark{% else %}bg-danger{% endif %}">

          {{ r.status }}

        </span>

      </td>

    </tr>

    {% endfor %}

  </tbody>

</table>

{% endblock %}

home.html

{% extends "base.html" %}

{% block content %}

<div class="container mt-4">

  <h1>Welcome to Recon Portal</h1>

  <p class="lead">Manage your financial reconciliation efficiently with our intuitive tools.</p>

  <div class="row mt-4">

    <div class="col-md-4">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Dashboard</h5>

          <p class="card-text">View reconciliation status and financial summaries.</p>

          <a href="{{ url\_for('dashboard') }}" class="btn btn-primary">Go to Dashboard</a>

        </div>

      </div>

    </div>

    <div class="col-md-4">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Clients</h5>

          <p class="card-text">Manage your client list and details.</p>

          <a href="{{ url\_for('list\_clients') }}" class="btn btn-primary">Manage Clients</a>

        </div>

      </div>

    </div>

    <div class="col-md-4">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Bills</h5>

          <p class="card-text">Add and edit bills for your clients.</p>

          <a href="{{ url\_for('bills') }}" class="btn btn-primary">Manage Bills</a>

        </div>

      </div>

    </div>

    <div class="col-md-4">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Receipts</h5>

          <p class="card-text">Track receipts and payments.</p>

          <a href="{{ url\_for('receipts') }}" class="btn btn-primary">Manage Receipts</a>

        </div>

      </div>

    </div>

  </div>

  <div class="mt-5">

    <p>Current date and time: <span id="current-time">{{ current\_time.strftime('%I:%M %p IST on %B %d, %Y') }}</span></p>

  </div>

</div>

<script>

  function updateTime() {

    const now = new Date().toLocaleString('en-US', { timeZone: 'Asia/Kolkata', hour12: true, hour: 'numeric', minute: 'numeric', month: 'long', day: 'numeric', year: 'numeric' });

    document.getElementById('current-time').textContent = now.replace(' at', ' IST on');

  }

  updateTime(); // Initial call

  setInterval(updateTime, 1000); // Update every second

</script>

{% endblock %}

receipts.html

{% extends "base.html" %}

{% block content %}

<h3>Add Receipt</h3>

<form method="post" class="row g-2" id="receiptForm">

  <div class="col-md-3">

  <label class="form-label" for="clientCombobox">Client</label>

  <!-- Text input becomes an ARIA combobox -->

  <input

    id="clientCombobox"

    class="form-control mb-1"

    type="text"

    placeholder="Type to find client"

    role="combobox"

    aria-autocomplete="list"

    aria-haspopup="listbox"

    aria-expanded="false"

    aria-controls="clientListbox"

    aria-activedescendant=""

    autocomplete="off"

  >

  <!-- List of options becomes a listbox; keep your existing items -->

  <div id="clientListbox"

       role="listbox"

       class="list-group"

       style="max-height:180px; overflow:auto;"

       hidden>

    {% for c in clients %}

      <button type="button"

              class="list-group-item list-group-item-action"

              role="option"

              id="client-opt-{{ c.id }}"

              data-id="{{ c.id }}"

              aria-selected="false">

        {{ c.name }}

      </button>

    {% endfor %}

  </div>

  <!-- Hidden value submitted to server -->

  <input type="hidden" name="client\_id" id="clientHidden" required>

</div>

  <div class="col-md-3">

  <label class="form-label" for="billCombobox">Bill No</label>

  <!-- Text input behaves as a combobox -->

  <input

    id="billCombobox"

    class="form-control mb-1"

    type="text"

    placeholder="Type to find bill no"

    role="combobox"

    aria-autocomplete="list"

    aria-haspopup="listbox"

    aria-expanded="false"

    aria-controls="billListbox"

    aria-activedescendant=""

    autocomplete="off"

  >

  <!-- List of bill numbers as a listbox -->

  <div id="billListbox"

       role="listbox"

       class="list-group"

       style="max-height:180px; overflow:auto;"

       hidden>

    {% for bn, cid in bills\_for\_dropdown %}

      {# Use a safe id; loop.index ensures uniqueness #}

      <button type="button"

              class="list-group-item list-group-item-action"

              role="option"

              id="bill-opt-{{ loop.index }}"

              data-bill="{{ bn }}"

              aria-selected="false">

        {{ bn }}

      </button>

    {% endfor %}

  </div>

  <!-- Hidden field actually submitted to server -->

  <input type="hidden" name="bill\_no" id="billHidden" required>

</div>

  <div class="col-md-3">

    <label class="form-label"> Receipt Amount</label>

    <input type="number" step="0.01" name="paid\_amount" id="paidInput" class="form-control" value="0">

  </div>

  <div class="col-md-3">

    <label class="form-label">TDS Amount</label>

    <input type="number"  name="tds\_amt" id="tdsInput" class="form-control" value="0">

  </div>

  <div class="col-md-3">

    <label class="form-label"> Total Amount (auto)</label>

    <input type="number" id="collectionAuto" class="form-control" value="0" readonly>

  </div>

  <div class="col-md-3">

    <label class="form-label">Receipt Date</label>

    <input type="date" name="receipt\_date" class="form-control" required>

  </div>

  <div class="col-md-3">

    <label class="form-label">UTR Details</label>

    <input name="utr\_details" class="form-control">

  </div>

   <div class="col-md-3">

    <label class="form-label">Receipt Ref</label>

    <input name="receipt\_ref" class="form-control">

  </div>

  <div class="col-md-3">

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

    <input name="mode" class="form-control" placeholder="NEFT/IMPS/Cash/etc">

  </div>

  <div class="col-12">

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

    <input name="remarks" class="form-control">

  </div>

  <div class="col-12">

    <button class="btn btn-primary">Save Receipt</button>

  </div>

</form>

<hr>

<h3>All Receipts</h3>

<input id="recSearch" class="form-control mb-2" placeholder="Search by client or bill no or UTR">

<table class="table table-sm table-striped">

  <thead>

    <tr>

      <th>Date</th>

      <th>Receipt Ref</th>

      <th>Client</th>

      <th>Bill No</th>

      <th>Bill Amount</th>

      <th>TDS</th>

      <th>Collection</th>

      <th>Status</th>

      <th>UTR</th>

      <th>Mode</th>

      <th>Remark</th>

      <th></th>

      <th></th>

    </tr>

  </thead>

  <tbody id="recTable">

    {% for r, bill\_amount, status in receipts %}

    <tr>

      <td>{{ r.receipt\_date }}</td>

      <td>{{ r.receipt\_ref }}</td>

      <td>{{ r.client.name }}</td>

      <td>{{ r.bill\_no }}</td>

      <td>{{ bill\_amount is not none and ("%.2f"|format(bill\_amount)) or "-" }}</td>

      <td>{{ "%.2f"|format(r.tds\_amt or 0) }}</td>

      <td>{{ "%.2f"|format(r.collection\_amount) }}</td>

      <td>

        {% if status %}

          <span class="badge

            {% if status=='Paid' %}bg-success

            {% elif status=='Pending' %}bg-warning text-dark

            {% else %}bg-danger

            {% endif %}">

            {{ status }}

          </span>

        {% else %}

          -

        {% endif %}

      </td>

      <td>{{ r.utr\_details }}</td>

      <td>{{ r.mode }}</td>

      <td>

        {% set rmk = r.remarks or '' %}

        {% if rmk|length > 25 %}

          {{ rmk[:25] }}…

          <button type="button" class="btn btn-sm btn-link p-0"

                  data-bs-toggle="popover" data-bs-trigger="focus"

                  title="Remarks" data-bs-content="{{ rmk|e }}">

            view

          </button>

        {% else %}

          {{ rmk }}

        {% endif %}

      </td>

      <td>

        <a class="btn btn-sm btn-outline-primary" href="{{ url\_for('edit\_receipt', rid=r.id) }}">Edit</a>

      </td>

      <td>

        <form method="post" action="{{ url\_for('delete\_row', table='receipt', row\_id=r.id) }}"

              onsubmit="return confirm('Delete receipt?')">

          <button class="btn btn-sm btn-outline-danger">Delete</button>

        </form>

      </td>

    </tr>

    {% endfor %}

  </tbody>

</table>

<script>

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

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

    const q = recSearch.value.toLowerCase();

    document.querySelectorAll('#recTable tr').forEach(tr => {

      const text = tr.innerText.toLowerCase();

      tr.style.display = text.includes(q) ? '' : 'none';

    });

  });

  // auto compute collection amount = TDS + Paid

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

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

  const collAuto = document.getElementById('collectionAuto');

  function updateCollection() {

    const t = parseFloat(tdsInput.value || 0);

    const p = parseFloat(paidInput.value || 0);

    collAuto.value = (t + p).toFixed(2);

  }

  tdsInput.addEventListener('input', updateCollection);

  paidInput.addEventListener('input', updateCollection);

  updateCollection();

</script>

<!-- Enable Bootstrap popovers -->

<script>

  document.querySelectorAll('[data-bs-toggle="popover"]').forEach(el=>{

    new bootstrap.Popover(el, {container:'body', html:false});

  });

</script>

<!-- Style tweaks for the picker lists (optional but recommended) -->

<style>

  #receiptClientList, #receiptBillList { max-height: 180px; overflow: auto; }

  #receiptClientList .list-group-item:hover,

  #receiptBillList .list-group-item:hover { background: #f2f6ff; }

</style>

<!-- Client picker: live filter + click to select -->

<script>

(function(){

  const input = document.getElementById('receiptClientFilter');

  const list  = document.getElementById('receiptClientList');

  const hid   = document.getElementById('receiptClientId');

  if (!input || !list || !hid) return;

  // Live filter

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

    const q = (input.value || '').toLowerCase();

    list.querySelectorAll('.list-group-item').forEach(btn => {

      const t = btn.textContent.toLowerCase();

      btn.style.display = t.includes(q) ? '' : 'none';

    });

  });

  // Click to select

  list.addEventListener('click', e => {

    const btn = e.target.closest('.list-group-item');

    if (!btn) return;

    hid.value = btn.dataset.id;            // value submitted to server

    input.value = btn.textContent.trim();  // show chosen client in the input

    list.querySelectorAll('.list-group-item').forEach(b => b.style.display = '');

  });

})();

</script>

<!-- Bill No picker: live filter + click to select -->

<script>

(function(){

  const input = document.getElementById('receiptBillFilter');

  const list  = document.getElementById('receiptBillList');

  const hid   = document.getElementById('receiptBillNo');

  if (!input || !list || !hid) return;

  // Live filter

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

    const q = (input.value || '').toLowerCase();

    list.querySelectorAll('.list-group-item').forEach(btn => {

      const t = btn.textContent.toLowerCase();

      btn.style.display = t.includes(q) ? '' : 'none';

    });

  });

  // Click to select

  list.addEventListener('click', e => {

    const btn = e.target.closest('.list-group-item');

    if (!btn) return;

    hid.value = btn.dataset.bill;          // value submitted to server

    input.value = btn.textContent.trim();  // show chosen bill no in the input

    list.querySelectorAll('.list-group-item').forEach(b => b.style.display = '');

  });

})();

</script>

<script>

(function(){

  const input = document.getElementById('clientCombobox');

  const list  = document.getElementById('clientListbox');

  const hid   = document.getElementById('clientHidden');

  if (!input || !list || !hid) return;

  let activeIndex = -1;

  let visible = []; // cache of visible options (buttons)

  function optionsAll() {

    return Array.from(list.querySelectorAll('[role="option"]'));

  }

  function openList() {

    list.hidden = false;

    input.setAttribute('aria-expanded', 'true');

    rebuildVisible();

  }

  function closeList() {

    list.hidden = true;

    input.setAttribute('aria-expanded', 'false');

    setActive(-1);

  }

  function setActive(i) {

    const opts = visible;

    // clear previous

    opts.forEach(o => o.classList.remove('active'));

    activeIndex = i;

    if (i >= 0 && i < opts.length) {

      const el = opts[i];

      el.classList.add('active');

      input.setAttribute('aria-activedescendant', el.id);

      // ensure visibility

      const r = el.getBoundingClientRect();

      const lr = list.getBoundingClientRect();

      if (r.top < lr.top) list.scrollTop -= (lr.top - r.top);

      if (r.bottom > lr.bottom) list.scrollTop += (r.bottom - lr.bottom);

    } else {

      input.removeAttribute('aria-activedescendant');

    }

  }

  function rebuildVisible() {

    const q = (input.value || '').toLowerCase();

    visible = [];

    optionsAll().forEach(btn => {

      const show = btn.textContent.toLowerCase().includes(q);

      btn.style.display = show ? '' : 'none';

      if (show) visible.push(btn);

      btn.setAttribute('aria-selected', 'false');

    });

    setActive(visible.length ? 0 : -1);

  }

  function commitSelection(btn) {

    if (!btn) return;

    // write to hidden and input

    hid.value = btn.dataset.id;

    input.value = btn.textContent.trim();

    // mark selected for SRs

    visible.forEach(o => o.setAttribute('aria-selected', o === btn ? 'true' : 'false'));

    closeList();

  }

  // Typing opens and filters

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

    openList();

    rebuildVisible();

  });

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

    openList();

  });

  input.addEventListener('blur', (e) => {

    // Small delay to allow click on option

    setTimeout(() => {

      if (!list.contains(document.activeElement)) closeList();

    }, 120);

  });

  // Keyboard navigation

  input.addEventListener('keydown', (e) => {

    if (e.key === 'ArrowDown') {

      e.preventDefault();

      if (list.hidden) openList();

      if (visible.length) setActive(Math.min(activeIndex + 1, visible.length - 1));

    } else if (e.key === 'ArrowUp') {

      e.preventDefault();

      if (visible.length) setActive(Math.max(activeIndex - 1, 0));

    } else if (e.key === 'Enter') {

      if (!list.hidden && activeIndex >= 0) {

        e.preventDefault();

        commitSelection(visible[activeIndex]);

      }

    } else if (e.key === 'Escape') {

      closeList();

    }

  });

  // Mouse selection

  list.addEventListener('mousedown', (e) => {

    // prevent input blur before we handle click

    e.preventDefault();

  });

  list.addEventListener('click', (e) => {

    const btn = e.target.closest('[role="option"]');

    if (btn) commitSelection(btn);

  });

})();

</script>

<style>

  #billListbox .list-group-item:hover { background:#f2f6ff; }

</style>

<!-- Bill No combobox (JS) — place near the bottom of the same template, after other scripts -->

<script>

(function(){

  const input = document.getElementById('billCombobox');

  const list  = document.getElementById('billListbox');

  const hid   = document.getElementById('billHidden');

  if (!input || !list || !hid) return;

  let activeIndex = -1;

  let visible = [];

  function optionsAll() {

    return Array.from(list.querySelectorAll('[role="option"]'));

  }

  function openList() {

    list.hidden = false;

    input.setAttribute('aria-expanded', 'true');

    rebuildVisible();

  }

  function closeList() {

    list.hidden = true;

    input.setAttribute('aria-expanded', 'false');

    setActive(-1);

  }

  function setActive(i) {

    const opts = visible;

    opts.forEach(o => o.classList.remove('active'));

    activeIndex = i;

    if (i >= 0 && i < opts.length) {

      const el = opts[i];

      el.classList.add('active');

      input.setAttribute('aria-activedescendant', el.id);

      const r = el.getBoundingClientRect();

      const lr = list.getBoundingClientRect();

      if (r.top < lr.top) list.scrollTop -= (lr.top - r.top);

      if (r.bottom > lr.bottom) list.scrollTop += (r.bottom - lr.bottom);

    } else {

      input.removeAttribute('aria-activedescendant');

    }

  }

  function rebuildVisible() {

    const q = (input.value || '').toLowerCase();

    visible = [];

    optionsAll().forEach(btn => {

      const show = btn.textContent.toLowerCase().includes(q);

      btn.style.display = show ? '' : 'none';

      if (show) visible.push(btn);

      btn.setAttribute('aria-selected', 'false');

    });

    setActive(visible.length ? 0 : -1);

  }

  function commitSelection(btn) {

    if (!btn) return;

    hid.value = btn.dataset.bill;           // submit this bill number

    input.value = btn.textContent.trim();   // show the selected bill in the input

    visible.forEach(o => o.setAttribute('aria-selected', o === btn ? 'true' : 'false'));

    closeList();

  }

  // Typing opens and filters

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

    openList();

    rebuildVisible();

  });

  input.addEventListener('focus', openList);

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

    // give time for click selection

    setTimeout(() => {

      if (!list.contains(document.activeElement)) closeList();

    }, 120);

  });

  // Keyboard navigation

  input.addEventListener('keydown', (e) => {

    if (e.key === 'ArrowDown') {

      e.preventDefault();

      if (list.hidden) openList();

      if (visible.length) setActive(Math.min(activeIndex + 1, visible.length - 1));

    } else if (e.key === 'ArrowUp') {

      e.preventDefault();

      if (visible.length) setActive(Math.max(activeIndex - 1, 0));

    } else if (e.key === 'Enter') {

      if (!list.hidden && activeIndex >= 0) {

        e.preventDefault();

        commitSelection(visible[activeIndex]);

      }

    } else if (e.key === 'Escape') {

      closeList();

    }

  });

  // Mouse selection

  list.addEventListener('mousedown', (e) => e.preventDefault()); // prevent blur before click

  list.addEventListener('click', (e) => {

    const btn = e.target.closest('[role="option"]');

    if (btn) commitSelection(btn);

  });

})();

</script>

{% endblock %}

app.py

from datetime import datetime

from dateutil.parser import parse as dateparse

from flask import Flask, render\_template, request, redirect, url\_for, flash, send\_file

from flask\_sqlalchemy import SQLAlchemy

import pandas as pd

import io

from pyngrok import ngrok

from flask\_migrate import Migrate

import pytz

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

app.secret\_key = "change-this-secret-key"

app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///data.db"

app.config["SQLALCHEMY\_TRACK\_MODIFICATIONS"] = False

db = SQLAlchemy(app)

migrate = Migrate(app, db)

# @app.route("/")

# def home():

#     # Get the ngrok public URL if available

#     public\_url = None

#     try:

#         tunnels = ngrok.get\_tunnels()

#         if tunnels:

#             public\_url = tunnels[0].public\_url

#     except Exception:

#         public\_url = None

#     return render\_template("dashboard.html", public\_url=public\_url)

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

# # NEW: Open an ngrok tunnel on port 5000

#     public\_url = ngrok.connect(5000)

#     print("Public URL:", public\_url)  # Share this link

#     # Run Flask on all interfaces so ngrok can access it

#     app.run(host="0.0.0.0", port=5000)

class Bill(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    bill\_no = db.Column(db.String(100), unique=True, nullable=False, index=True)

    bill\_date = db.Column(db.Date, nullable=False)

    client\_id = db.Column(db.Integer, db.ForeignKey('client.id'), nullable=False, index=True)

    amount = db.Column(db.Float, nullable=False)

    description = db.Column(db.String(50000), nullable=True)

    remarks = db.Column(db.String(50000), nullable=True)

    Subject = db.Column(db.String(255), nullable=True)

    client = db.relationship('Client', backref=db.backref('bills', lazy=True))

class Receipt(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    receipt\_ref = db.Column(db.String(100), nullable=True, index=True)  # optional external reference

    receipt\_date = db.Column(db.Date, nullable=False)

    client\_id = db.Column(db.Integer, db.ForeignKey('client.id'), nullable=False, index=True)

    bill\_no = db.Column(db.String(100), nullable=True, index=True)  # match a Bill if provided

    tds\_amt = db.Column(db.Float, nullable=True, default=0.0)

    collection\_amount = db.Column(db.Float, nullable=False)  # Receipt/Collection Amount

    utr\_details = db.Column(db.String(255), nullable=True)

    mode = db.Column(db.String(100), nullable=True)

    remarks = db.Column(db.String(10000), nullable=True)

    client = db.relationship('Client', backref=db.backref('receipts', lazy=True))

class Client(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    name = db.Column(db.String(100), nullable=False, unique=True)

    address = db.Column(db.String(200))

    gst\_no = db.Column(db.String(500))

    pan\_no = db.Column(db.String(500))

    remarks = db.Column(db.Text)

def parse\_date(s, default=None):

    if not s:

        return default

    try:

        return dateparse(s, dayfirst=True).date()

    except Exception:

        return default

# app.py (routes) — UPDATE/REPLACE as indicated

@app.route("/")

def home():

    ist = pytz.timezone('Asia/Kolkata')

    current\_time = datetime.now(ist)

    return render\_template("home.html", current\_time=current\_time)

@app.route("/")

def index():

    return redirect(url\_for("dashboard"))

# Clients

@app.route("/clients", methods=["GET", "POST"])

def clients():

    if request.method == "POST":

        name = request.form.get("name", "").strip()

        address = request.form.get("address", "").strip()

        gst\_no = request.form.get("gst\_no", "").strip()

        pan\_no = request.form.get("pan\_no", "").strip()

        remarks = request.form.get("remarks", "").strip()

        if not name:

            flash("Client name cannot be empty.", "danger")

        elif Client.query.filter(db.func.lower(Client.name) == name.lower()).first():

            flash("Client already exists.", "danger")

        else:

            new\_client = Client(

                name=name,

                address=address,

                gst\_no=gst\_no,

                pan\_no=pan\_no,

                remarks=remarks

            )

            db.session.add(new\_client)

            db.session.commit()

            flash("Client added successfully.", "success")

        return redirect(url\_for("clients"))

    all\_clients = Client.query.order\_by(Client.name.asc()).all()

    return render\_template("clients.html", clients=all\_clients)

# Bills

@app.route("/bills", methods=["GET", "POST"])

def bills():

    clients = Client.query.order\_by(Client.name.asc()).all()

    if request.method == "POST":

        bill\_no = request.form.get("bill\_no", "").strip()

        bill\_date\_str = request.form.get("bill\_date")

        client\_id = request.form.get("client\_id")

        amount = float(request.form.get("amount") or 0)

        description = request.form.get("description", "").strip()

        remarks = request.form.get("remarks", "").strip()

        Subject = request.form.get("Subject", "").strip()

        if not bill\_no or not bill\_date\_str or not client\_id or amount <= 0:

            flash("Please fill Bill No, Date, Client and positive Amount.", "danger")

        elif Bill.query.filter\_by(bill\_no=bill\_no).first():

            flash("Bill No already exists.", "danger")

        else:

            bill\_date = parse\_date(bill\_date\_str)

            db.session.add(Bill(

                bill\_no=bill\_no, bill\_date=bill\_date, client\_id=int(client\_id),

                amount=amount, description=description, remarks=remarks

            ))

            db.session.commit()

            flash("Bill saved.", "success")

        return redirect(url\_for("bills"))

    bills = Bill.query.order\_by(Bill.bill\_date.desc(), Bill.id.desc()).all()

    return render\_template("bills.html", bills=bills, clients=clients)

# app.py — NEW: edit route for bills

@app.route("/bills/<int:bid>/edit", methods=["GET", "POST"])

def edit\_bill(bid):

    b = Bill.query.get\_or\_404(bid)

    clients = Client.query.order\_by(Client.name.asc()).all()

    if request.method == "POST":

        new\_bill\_no = request.form.get("bill\_no", "").strip()

        if not new\_bill\_no:

            flash("Bill No is required.", "danger")

            return redirect(request.url)

        if new\_bill\_no != b.bill\_no and Bill.query.filter\_by(bill\_no=new\_bill\_no).first():

            flash("Bill No already exists.", "danger")

            return redirect(request.url)

        b.bill\_no = new\_bill\_no

        b.bill\_date = parse\_date(request.form.get("bill\_date"))

        b.client\_id = int(request.form.get("client\_id"))

        b.amount = float(request.form.get("amount") or 0)

        b.description = request.form.get("description", "").strip()

        b.remarks = request.form.get("remarks", "").strip()

        b.Subject = request.form.get("Subject", "").strip()

        db.session.commit()

        flash("Bill updated.", "success")

        return redirect(url\_for("bills"))

    return render\_template("bill\_edit.html", b=b, clients=clients)

# Receipts

# app.py — REPLACE the entire receipts() view with this

@app.route("/receipts", methods=["GET", "POST"])

def receipts():

    clients = Client.query.order\_by(Client.name.asc()).all()

    bills\_for\_dropdown = Bill.query.with\_entities(Bill.bill\_no, Bill.client\_id).order\_by(Bill.bill\_no.asc()).all()

    if request.method == "POST":

        receipt\_ref = request.form.get("receipt\_ref", "").strip()

        receipt\_date\_str = request.form.get("receipt\_date")

        client\_id = request.form.get("client\_id")

        bill\_no = request.form.get("bill\_no", "").strip()

        tds\_amt = float(request.form.get("tds\_amt") or 0)

        paid\_amount = float(request.form.get("paid\_amount") or 0)  # NEW

        collection\_amount = tds\_amt + paid\_amount                  # NEW

        utr\_details = request.form.get("utr\_details", "").strip()

        mode = request.form.get("mode", "").strip()

        remarks = request.form.get("remarks", "").strip()

        if not receipt\_date\_str or not client\_id or collection\_amount <= 0 or not bill\_no:

            flash("Please fill Receipt Date, Client, Bill No and positive amounts.", "danger")

        elif not Bill.query.filter\_by(bill\_no=bill\_no).first():

            flash("Selected Bill No does not exist.", "danger")

        else:

            receipt\_date = parse\_date(receipt\_date\_str)

            db.session.add(Receipt(

                receipt\_ref=receipt\_ref, receipt\_date=receipt\_date, client\_id=int(client\_id),

                bill\_no=bill\_no, tds\_amt=tds\_amt, collection\_amount=collection\_amount,

                utr\_details=utr\_details, mode=mode, remarks=remarks

            ))

            db.session.commit()

            flash("Receipt saved.", "success")

        return redirect(url\_for("receipts"))

    # GET: annotated rows with bill amount and status

    all\_receipts = Receipt.query.order\_by(Receipt.receipt\_date.desc(), Receipt.id.desc()).all()

    bills\_map = {b.bill\_no: b.amount for b in Bill.query.with\_entities(Bill.bill\_no, Bill.amount).all()}

    paid\_map = {}

    paid\_rows = (

        db.session.query(Receipt.bill\_no, db.func.sum(Receipt.collection\_amount))

        .filter(Receipt.bill\_no.isnot(None))

        .group\_by(Receipt.bill\_no)

        .all()

    )

    for bill\_no\_x, total in paid\_rows:

        paid\_map[bill\_no\_x] = float(total or 0)

    annotated = []

    for r in all\_receipts:

        bill\_amount = bills\_map.get(r.bill\_no) if r.bill\_no else None

        paid\_total = paid\_map.get(r.bill\_no, 0.0) if r.bill\_no else None

        status = None

        if bill\_amount is not None:

            balance = bill\_amount - (paid\_total or 0.0)

            status = "Paid" if abs(balance) < 0.0001 else ("Overpaid" if balance < 0 else "Pending")

        annotated.append((r, bill\_amount, status))

    return render\_template(

        "receipts.html",

        receipts=annotated,

        clients=clients,

        bills\_for\_dropdown=bills\_for\_dropdown

    )

# app.py — NEW: edit route for receipts

@app.route("/receipts/<int:rid>/edit", methods=["GET", "POST"])

def edit\_receipt(rid):

    r = Receipt.query.get\_or\_404(rid)

    clients = Client.query.order\_by(Client.name.asc()).all()

    if request.method == "POST":

        r.receipt\_ref = request.form.get("receipt\_ref", "").strip()

        r.receipt\_date = parse\_date(request.form.get("receipt\_date"))

        r.client\_id = int(request.form.get("client\_id"))

        r.bill\_no = request.form.get("bill\_no", "").strip()

        tds\_amt = float(request.form.get("tds\_amt") or 0)

        paid\_amount = float(request.form.get("paid\_amount") or 0)

        r.tds\_amt = tds\_amt

        r.collection\_amount = tds\_amt + paid\_amount

        r.utr\_details = request.form.get("utr\_details", "").strip()

        r.mode = request.form.get("mode", "").strip()

        r.remarks = request.form.get("remarks", "").strip()

        if not r.bill\_no or not Bill.query.filter\_by(bill\_no=r.bill\_no).first():

            flash("Valid Bill No is required.", "danger")

            return redirect(request.url)

        db.session.commit()

        flash("Receipt updated.", "success")

        return redirect(url\_for("receipts"))

    paid\_amount\_current = (r.collection\_amount or 0) - (r.tds\_amt or 0)

    bills\_for\_dropdown = Bill.query.with\_entities(Bill.bill\_no, Bill.client\_id).order\_by(Bill.bill\_no.asc()).all()

    return render\_template("receipt\_edit.html", r=r, clients=clients, paid\_amount\_current=paid\_amount\_current, bills\_for\_dropdown=bills\_for\_dropdown)

# Delete

@app.route("/delete/<table>/<int:row\_id>", methods=["POST"])

def delete\_row(table, row\_id):

    model = Bill if table == "bill" else Receipt if table == "receipt" else Client if table == "client" else None

    if not model:

        flash("Invalid delete request.", "danger")

        return redirect(url\_for("index"))

    rec = model.query.get(row\_id)

    if rec:

        db.session.delete(rec)

        db.session.commit()

        flash("Deleted.", "success")

    return redirect(request.referrer or url\_for("dashboard"))

# REMOVE any earlier @app.route("/clients", methods=["GET","POST"]) def clients(): ...

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

def list\_clients():

    all\_clients = Client.query.order\_by(Client.name.asc()).all()

    return render\_template("clients.html", clients=all\_clients)

@app.route("/clients", methods=["POST"])

def add\_client():

    name = (request.form.get("name") or "").strip()

    if not name:

        flash("Client name cannot be empty.", "danger")

        return redirect(url\_for("list\_clients"))

    exists = Client.query.filter(db.func.lower(Client.name) == name.lower()).first()

    if exists:

        flash("Client already exists.", "danger")

        return redirect(url\_for("list\_clients"))

    new\_client = Client(

        name=name,

        address=(request.form.get("address") or "").strip(),

        gst\_no=(request.form.get("gst\_no") or "").strip(),

        pan\_no=(request.form.get("pan\_no") or "").strip(),

        remarks=(request.form.get("remarks") or "").strip()

    )

    try:

        db.session.add(new\_client)

        db.session.commit()

        flash("Client added successfully.", "success")

    except Exception as e:

        db.session.rollback()

        flash(f"Could not add client: {str(e)}", "danger")

    return redirect(url\_for("list\_clients"))

@app.route("/clients/<int:cid>/edit", methods=["GET", "POST"])

def edit\_client(cid):

    client = Client.query.get\_or\_404(cid)

    if request.method == "POST":

        client.name = (request.form.get("name") or "").strip()

        client.address = (request.form.get("address") or "").strip()

        client.gst\_no = (request.form.get("gst\_no") or "").strip()

        client.pan\_no = (request.form.get("pan\_no") or "").strip()

        client.remarks = (request.form.get("remarks") or "").strip()

        try:

            db.session.commit()

            flash("Client updated", "success")

            return redirect(url\_for("list\_clients"))

        except Exception:

            db.session.rollback()

            flash("Could not update client", "danger")

    return render\_template("client\_edit.html", client=client)

# Dashboard / Reconciliation

@app.route("/dashboard")

def dashboard():

    client\_q = request.args.get("client", "").strip()

    status = request.args.get("status", "").strip()  # "", Paid, Pending, Overpaid

    df\_str = request.args.get("from", "").strip()

    dt\_str = request.args.get("to", "").strip()

    dfrom = parse\_date(df\_str, default=None)

    dto = parse\_date(dt\_str, default=None)

    # Build queries

    bq = Bill.query

    rq = Receipt.query

    if client\_q:

        bq = bq.join(Client).filter(Client.name.ilike(f"%{client\_q}%"))

        rq = rq.join(Client).filter(Client.name.ilike(f"%{client\_q}%"))

    if dfrom:

        bq = bq.filter(Bill.bill\_date >= dfrom)

        rq = rq.filter(Receipt.receipt\_date >= dfrom)

    if dto:

        bq = bq.filter(Bill.bill\_date <= dto)

        rq = rq.filter(Receipt.receipt\_date <= dto)

    # DataFrames for reconciliation

    bills\_df = pd.read\_sql(bq.statement, db.engine)

    receipts\_df = pd.read\_sql(rq.statement, db.engine)

    # Map client\_id -> name for display

    clients\_map = {c.id: c.name for c in Client.query.all()}

    if not bills\_df.empty:

        bills\_df["client\_name"] = bills\_df["client\_id"].map(clients\_map)

    if not receipts\_df.empty:

        receipts\_df["client\_name"] = receipts\_df["client\_id"].map(clients\_map)

    # Sum receipts by bill\_no

    pay\_by\_bill = (

        receipts\_df.dropna(subset=["bill\_no"])

        .groupby("bill\_no")["collection\_amount"]

        .sum()

        .reset\_index()

        .rename(columns={"collection\_amount": "paid\_amount"})

        if not receipts\_df.empty else pd.DataFrame(columns=["bill\_no","paid\_amount"])

    )

    recon = bills\_df.merge(pay\_by\_bill, how="left", on="bill\_no")

    if recon.empty:

        recon = bills\_df.copy()

        recon["paid\_amount"] = 0.0

    recon["paid\_amount"] = recon["paid\_amount"].fillna(0.0)

    recon["balance"] = recon["amount"] - recon["paid\_amount"]

    recon["status"] = recon.apply(

        lambda r: "Paid" if abs(r["balance"]) < 0.0001 else ("Overpaid" if r["balance"] < 0 else "Pending"), axis=1

    )

    if status:

        recon = recon[recon["status"].str.lower() == status.lower()]

    # Totals for dashboard cards

    totals = {

        "total\_bills": float(recon["amount"].sum()) if not recon.empty else 0.0,

        "total\_paid": float(recon["paid\_amount"].sum()) if not recon.empty else 0.0,

        "total\_balance": float(recon["balance"].sum()) if not recon.empty else 0.0,

        "count\_pending": int((recon["status"] == "Pending").sum()) if not recon.empty else 0,

        "count\_paid": int((recon["status"] == "Paid").sum()) if not recon.empty else 0,

        "count\_overpaid": int((recon["status"] == "Overpaid").sum()) if not recon.empty else 0,

    }

    # Sort by date, then bill\_no

    recon = recon.sort\_values(["bill\_date","bill\_no"], ascending=[False, True]) if not recon.empty else recon

    return render\_template(

        "dashboard.html",

        filters={"client": client\_q, "status": status, "dfrom": df\_str, "dto": dt\_str},

        rows=recon.to\_dict(orient="records"),

        totals=totals,

        clients=Client.query.order\_by(Client.name.asc()).all()

        )

# app.py — add this JSON endpoint (after your other routes)

@app.get("/api/bills/by-client/<int:client\_id>")

def api\_bills\_by\_client(client\_id: int):

    rows = Bill.query.with\_entities(Bill.bill\_no).filter(Bill.client\_id == client\_id).order\_by(Bill.bill\_no.asc()).all()

    return {"bills": [bn for (bn,) in rows]}

# Export (CSV/XLSX)

@app.route("/export/reconciliation.<fmt>")

def export\_reconciliation(fmt):

    bq = Bill.query

    rq = Receipt.query

    bills\_df = pd.read\_sql(bq.statement, db.engine)

    receipts\_df = pd.read\_sql(rq.statement, db.engine)

    clients\_map = {c.id: c.name for c in Client.query.all()}

    if not bills\_df.empty:

        bills\_df["client\_name"] = bills\_df["client\_id"].map(clients\_map)

    if not receipts\_df.empty:

        receipts\_df["client\_name"] = receipts\_df["client\_id"].map(clients\_map)

    pay\_by\_bill = (

        receipts\_df.dropna(subset=["bill\_no"])

        .groupby("bill\_no")["collection\_amount"]

        .sum()

        .reset\_index()

        .rename(columns={"collection\_amount": "paid\_amount"})

        if not receipts\_df.empty else pd.DataFrame(columns=["bill\_no","paid\_amount"])

    )

    recon = bills\_df.merge(pay\_by\_bill, how="left", on="bill\_no")

    if recon.empty:

        recon = bills\_df.copy()

        recon["paid\_amount"] = 0.0

    recon["paid\_amount"] = recon["paid\_amount"].fillna(0.0)

    recon["balance"] = recon["amount"] - recon["paid\_amount"]

    recon["status"] = recon.apply(

        lambda r: "Paid" if abs(r["balance"]) < 0.0001 else ("Overpaid" if r["balance"] < 0 else "Pending"), axis=1

    )

    output = io.BytesIO()

    if fmt == "csv":

        recon.to\_csv(output, index=False)

        output.seek(0)

        return send\_file(output, mimetype="text/csv", as\_attachment=True, download\_name="reconciliation.csv")

    elif fmt == "xlsx":

        with pd.ExcelWriter(output, engine="openpyxl") as writer:

            recon.to\_excel(writer, sheet\_name="Reconciliation", index=False)

        output.seek(0)

        return send\_file(

            output,

            mimetype="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",

            as\_attachment=True,

            download\_name="reconciliation.xlsx"

        )

    else:

        flash("Unsupported export format.", "danger")

        return redirect(url\_for("dashboard"))

with app.app\_context():

    db.create\_all()

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

    app.run(debug=True)