Reports & Automation Boilerplate for HostPilotPro

This document contains a ready-to-run **React (Vite)** frontend + **Express** backend boilerplate for the **Reports & Analytics** and **Automation & Alerts** modules. Paste these files into your Replit `/property-hub` project and follow the install/run steps below.

Project structure (suggested)

```
/property-hub
├── backend
│   ├── package.json
│   ├── server.js
│   ├── db.json             <-- lowdb storage (auto-created)
│   └── routes
│       ├── reports.js
│       └── automations.js
└── frontend
    ├── package.json
    ├── index.html
    └── src
        ├── main.jsx
        ├── App.jsx
        ├── pages
        │   ├── Reports.jsx
        │   └── Automation.jsx
        └── api.js
```

Backend

Install (backend folder)

```
cd backend
npm init -y
npm install express lowdb cors body-parser json2csv node-cron jspdf pdfkit
```

`lowdb` will give a tiny JSON DB for demo. Replace with your real DB (Postgres/MySQL) in production.

backend/server.js

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const reportsRouter = require('./routes/reports');
const automationsRouter = require('./routes/automations');

const app = express();
app.use(cors());
app.use(bodyParser.json());

app.use('/api/reports', reportsRouter);
app.use('/api/automations', automationsRouter);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(`Backend running on ${PORT}`));
```

backend/routes/reports.js

```
const express = require('express');
const { Low, JSONFile } = require('lowdb');
const { Parser } = require('json2csv');
const fs = require('fs');
const path = require('path');

const router = express.Router();
const file = path.join(__dirname, '..', 'db.json');
const adapter = new JSONFile(file);
const db = new Low(adapter);

// Ensure DB initial structure
(async () => { await db.read(); db.data = db.data || { properties: [],
bookings: [], finances: [], tasks: [], automations: [] }; await
db.write(); })();

// GET aggregated summary
router.get('/summary', async (req, res) => {
  await db.read();
  const { bookings = [], finances = [], tasks = [] } = db.data;

  // simple aggregations
  const totalBookings = bookings.length;
  const totalIncome = finances.filter(f => f.type === 'income').reduce((s,
f) => s + (f.amount||0), 0);
  const totalExpense = finances.filter(f => f.type === 'expense').reduce((s,
f) => s + (f.amount||0), 0);
  const taskCompletionRate = tasks.length ?
(tasks.filter(t=>t.completed).length / tasks.length) * 100 : 0;
```

```
  // bookings by month (basic)
  const byMonth = {};
  bookings.forEach(b => {
    const m = new Date(b.date).toISOString().slice(0,7);
    byMonth[m] = (byMonth[m]||0) + 1;
  });

  res.json({ totalBookings, totalIncome, totalExpense, taskCompletionRate,
bookingsByMonth: byMonth });
});

// POST export CSV
router.post('/export', async (req, res) => {
  await db.read();
  const { type = 'bookings' } = req.body; // bookings, finances, tasks,
properties
  const data = db.data[type] || [];

  try {
    const parser = new Parser();
    const csv = parser.parse(data);
    res.setHeader('Content-disposition', `attachment; filename=${type}.csv`);
    res.set('Content-Type', 'text/csv');
    return res.status(200).send(csv);
  } catch (err) {
    return res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

backend/routes/automations.js

```
const express = require('express');
const { Low, JSONFile } = require('lowdb');
const path = require('path');

const router = express.Router();
const file = path.join(__dirname, '..', 'db.json');
const adapter = new JSONFile(file);
const db = new Low(adapter);

(async () => { await db.read(); db.data = db.data || { automations: [] };
await db.write(); })();

// list
router.get('/', async (req, res) => {
  await db.read();
  res.json(db.data.automations || []);
```

```
});

// create
router.post('/', async (req, res) => {
  await db.read();
  const rule = req.body;
  rule.id = Date.now().toString();
  db.data.automations.push(rule);
  await db.write();
  res.status(201).json(rule);
});

// update
router.put('/:id', async (req, res) => {
  await db.read();
  const id = req.params.id;
  const idx = (db.data.automations||[]).findIndex(a => a.id === id);
  if (idx === -1) return res.status(404).send('Not found');
  db.data.automations[idx] = { ...db.data.automations[idx], ...req.body };
  await db.write();
  res.json(db.data.automations[idx]);
});

// delete
router.delete('/:id', async (req, res) => {
  await db.read();
  db.data.automations = (db.data.automations||[]).filter(a => a.id !==
req.params.id);
  await db.write();
  res.status(204).send();
});

module.exports = router;
```

Frontend (Vite + React)

Install (frontend folder)

```
cd frontend
npm init vite@latest . --template react
npm install
npm install axios chart.js react-chartjs-2 json2csv jspdf
```

`frontend/src/api.js`

```
import axios from 'axios';
const BASE = import.meta.env.VITE_API_URL || 'http://localhost:4000';
export const api = axios.create({ baseURL: BASE + '/api' });
```

Make sure Vite env `VITE_API_URL` points to your Replit backend or use relative proxy in dev.

`frontend/src/pages/Reports.jsx`

```
import React, { useEffect, useState } from 'react';
import { api } from '../api';
import { Line } from 'react-chartjs-2';
import { saveAs } from 'file-saver';

export default function Reports(){
  const [summary, setSummary] = useState(null);
  useEffect(()=>{ api.get('/reports/
summary').then(r=>setSummary(r.data)).catch(console.error) },[]);

  const handleExport = (type) => {
    api.post('/reports/export', { type }, { responseType: 'blob' })
      .then(r => {
        saveAs(r.data, `${type}.csv`);
      })
      .catch(console.error);
  }

  if(!summary) return <div>Loading...</div>;

  const labels = Object.keys(summary.bookingsByMonth || {});
  const data = Object.values(summary.bookingsByMonth || {});

  return (
    <div>
      <h2>Reports & Analytics</h2>
      <div>Total Bookings: {summary.totalBookings}</div>
      <div>Total Income: {summary.totalIncome}</div>
      <div>Total Expense: {summary.totalExpense}</div>
      <div>Task Completion: {summary.taskCompletionRate.toFixed(1)}%</div>

      <div style={{width:600}}>
        <Line data={{ labels, datasets:[{ label:'Bookings', data }] }} />
      </div>

      <button onClick={()=>handleExport('bookings')}>Export Bookings CSV</
button>
      <button onClick={()=>handleExport('finances')}>Export Finances CSV</
button>
```

```
    </div>
  );
}
```

frontend/src/pages/Automation.jsx

```
import React, { useEffect, useState } from 'react';
import { api } from '../api';

export default function Automation(){
  const [rules, setRules] = useState([]);
  const [form, setForm] = useState({name:'', trigger:'', action:''});

  useEffect(()=>{ api.get('/
automations').then(r=>setRules(r.data)).catch(console.error) },[]);

  const create = async () => {
    const res = await api.post('/automations', form);
    setRules(prev=>[...prev, res.data]);
    setForm({name:'', trigger:'', action:''});
  }

  const remove = async (id) => { await api.delete(`/automations/${id}`);
setRules(prev=>prev.filter(r=>r.id!==id)); }

  return (
    <div>
      <h2>Automation & Alerts</h2>
      <p>Define simple rules to schedule reminders and notifications.</p>

      <div>
        <input placeholder="Name" value={form.name}
onChange={e=>setForm({...form,name:e.target.value})} />
        <input placeholder="Trigger (e.g. booking_due)" value={form.trigger}
onChange={e=>setForm({...form,trigger:e.target.value})} />
        <input placeholder="Action (e.g. email_reminder)"
value={form.action} onChange={e=>setForm({...form,action:e.target.value})} />
        <button onClick={create}>Create Rule</button>
      </div>

      <ul>
        {rules.map(r=> (
          <li key={r.id}>{r.name} — {r.trigger} — {r.action} <button
onClick={()=>remove(r.id)}>Delete</button></li>
        ))}
      </ul>
    </div>
  );
}
```

Wire-up UI card clicks

Update the dashboard card components (where `Coming Soon` buttons are) to navigate:

```
// Example inside dashboard card
<button onClick={() => navigate('/reports')}>Reports & Analytics</button>
<button onClick={() => navigate('/automation')}>Automation & Alerts</button>
```

Remove the disabled/greyed-out styles after wiring the route.

Running locally on Replit

1. Create two repl services or one monorepo and run concurrently.
2. Start backend: `node server.js` (ensure port 4000 exposed).
3. Start frontend (Vite): `npm run dev` (serve on port 3000 or default). Set `VITE_API_URL` to your backend URL.

Next steps & enhancements

- Replace `lowdb` with your production DB and adjust queries.
- Add authentication and scoped report permissions.
- Add scheduled worker (cron) that uses automations rules to send emails (use SendGrid) or push notifications.
- Integrate AI insights: create an endpoint `/api/reports/insights` that calls an LLM with aggregated metrics to produce short textual insights.

If you want, I can now: - Generate the exact files as downloadable files inside the canvas (or a ZIP). - Create a concise Replit-agent script that executes the full setup (file creation, dependency installation, and start commands) automatically in one go. The script should: - Navigate to `/property-hub` - Run the `setup_replit_agent.sh` setup file if it exists, or create it if missing - Install dependencies for both backend and frontend - Start backend using `node backend/server.js` in background - Start frontend using `npm --prefix frontend run dev` so the dashboard is served immediately

Tell me which you prefer and I will add it to the canvas.

Replit agent setup script (creates files & installs dependencies)

A script named `setup_replit_agent.sh` has been added below. **Do not paste this script elsewhere** — run it from the `/property-hub` project root in your Replit shell. It will create the backend/frontend files described in this document and run npm installs.

How to run (on Replit): 1. Open the Replit shell at the project root (`/property-hub`). 2. Make script executable: `chmod +x setup_replit_agent.sh` . 3. Run it: `./setup_replit_agent.sh` .

What it does: - Creates `backend/` and `frontend/` folders and writes server and frontend source files (Reports & Automation modules). - Installs required npm packages in both folders. - Adds `db.json` (lowdb) initial file. - Updates `frontend/package.json` and `backend/package.json` with start scripts.

> The script will not start the servers automatically (so Replit won't keep processes running unexpectedly). After it finishes, start backend with `node backend/server.js` and frontend with `npm --prefix frontend run dev` (or configure Replit run commands).

```
#!/usr/bin/env bash
set -e
ROOT_DIR=$(pwd)
echo "Running Replit setup script in $ROOT_DIR"

# Create backend
mkdir -p backend/routes
cat > backend/package.json <<'JSON'
{
  "name": "hostpilot-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  }
}
JSON

cat > backend/server.js <<'JS'
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const reportsRouter = require('./routes/reports');
const automationsRouter = require('./routes/automations');

const app = express();
app.use(cors());
app.use(bodyParser.json());

app.use('/api/reports', reportsRouter);
app.use('/api/automations', automationsRouter);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(`Backend running on ${PORT}`));
JS

cat > backend/routes/reports.js <<'JS'
const express = require('express');
```

```
const { Low, JSONFile } = require('lowdb');
const { Parser } = require('json2csv');
const fs = require('fs');
const path = require('path');

const router = express.Router();
const file = path.join(__dirname, '..', 'db.json');
const adapter = new JSONFile(file);
const db = new Low(adapter);

(async () => { await db.read(); db.data = db.data || { properties: [],
bookings: [], finances: [], tasks: [], automations: [] }; await
db.write(); })();

router.get('/summary', async (req, res) => {
  await db.read();
  const { bookings = [], finances = [], tasks = [] } = db.data;
  const totalBookings = bookings.length;
  const totalIncome = finances.filter(f => f.type === 'income').reduce((s,
f) => s + (f.amount||0), 0);
  const totalExpense = finances.filter(f => f.type === 'expense').reduce((s,
f) => s + (f.amount||0), 0);
  const taskCompletionRate = tasks.length ?
(tasks.filter(t=>t.completed).length / tasks.length) * 100 : 0;
  const byMonth = {};
  bookings.forEach(b => {
    const m = b.date ? new Date(b.date).toISOString().slice(0,7) : 'unknown';
    byMonth[m] = (byMonth[m]||0) + 1;
  });
  res.json({ totalBookings, totalIncome, totalExpense, taskCompletionRate,
bookingsByMonth: byMonth });
});

router.post('/export', async (req, res) => {
  await db.read();
  const { type = 'bookings' } = req.body;
  const data = db.data[type] || [];
  try {
    const parser = new Parser();
    const csv = parser.parse(data);
    res.setHeader('Content-disposition', `attachment; filename=${type}.csv`);
    res.set('Content-Type', 'text/csv');
    return res.status(200).send(csv);
  } catch (err) {
    return res.status(500).json({ error: err.message });
  }
});

module.exports = router;
JS
```

```
cat > backend/routes/automations.js <<'JS'
const express = require('express');
const { Low, JSONFile } = require('lowdb');
const path = require('path');

const router = express.Router();
const file = path.join(__dirname, '..', 'db.json');
const adapter = new JSONFile(file);
const db = new Low(adapter);

(async () => { await db.read(); db.data = db.data || { automations: [] };
await db.write(); })();

router.get('/', async (req, res) => { await db.read();
res.json(db.data.automations || []); });
router.post('/', async (req, res) => { await db.read(); const rule =
req.body; rule.id = Date.now().toString(); db.data.automations.push(rule);
await db.write(); res.status(201).json(rule); });
router.put('/:id', async (req, res) => { await db.read(); const id =
req.params.id; const idx = (db.data.automations||[]).findIndex(a => a.id ===
id); if (idx === -1) return res.status(404).send('Not found');
db.data.automations[idx] = { ...db.data.automations[idx], ...req.body };
await db.write(); res.json(db.data.automations[idx]); });
router.delete('/:id', async (req, res) => { await db.read();
db.data.automations = (db.data.automations||[]).filter(a => a.id !==
req.params.id); await db.write(); res.status(204).send(); });

module.exports = router;
JS

# db.json
cat > backend/db.json <<'JSON'
{
  "properties": [],
  "bookings": [],
  "finances": [],
  "tasks": [],
  "automations": []
}
JSON

# Install backend deps
(cd backend && npm install express lowdb cors body-parser json2csv)

# Create frontend
mkdir -p frontend/src/pages
cat > frontend/package.json <<'JSON'
{
  "name": "hostpilot-frontend",
  "version": "1.0.0",
  "private": true,
```

```
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  }
}
JSON

cat > frontend/index.html <<'HTML'
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>HostPilotPro - Frontend</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
HTML

cat > frontend/src/main.jsx <<'JS'
import React from 'react'
import { createRoot } from 'react-dom/client'
import App from './App'
import './style.css'

createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
JS

cat > frontend/src/App.jsx <<'JS'
import React from 'react'
import { BrowserRouter, Routes, Route, Link } from 'react-router-dom'
import Reports from './pages/Reports'
import Automation from './pages/Automation'

export default function App(){
  return (
    <BrowserRouter>
      <div style={{padding:20}}>
        <nav><Link to="/reports">Reports</Link> | <Link to="/
automation">Automation</Link></nav>
        <Routes>
          <Route path="/reports" element={<Reports/>} />
          <Route path="/automation" element={<Automation/>} />
```

```
          <Route path="/" element={<div>Open Reports or Automation</div>} />
        </Routes>
      </div>
    </BrowserRouter>
  )
}
JS

cat > frontend/src/api.js <<'JS'
import axios from 'axios';
const BASE = import.meta.env.VITE_API_URL || 'http://localhost:4000';
export const api = axios.create({ baseURL: BASE + '/api' });
JS

cat > frontend/src/pages/Reports.jsx <<'JS'
import React, { useEffect, useState } from 'react';
import { api } from '../api';
import { Line } from 'react-chartjs-2';

export default function Reports(){
  const [summary, setSummary] = useState(null);
  useEffect(()=>{ api.get('/reports/
summary').then(r=>setSummary(r.data)).catch(console.error) },[]);
  const handleExport = (type) => { api.post('/reports/export', { type },
{ responseType: 'blob' }).then(r => { const url =
window.URL.createObjectURL(r.data); const a = document.createElement('a');
a.href = url; a.download = `${type}.csv`;
a.click(); }).catch(console.error); }
  if(!summary) return <div>Loading...</div>;
  const labels = Object.keys(summary.bookingsByMonth || {});
  const data = Object.values(summary.bookingsByMonth || {});
  return (
    <div>
      <h2>Reports & Analytics</h2>
      <div>Total Bookings: {summary.totalBookings}</div>
      <div>Total Income: {summary.totalIncome}</div>
      <div>Total Expense: {summary.totalExpense}</div>
      <div>Task Completion: {summary.taskCompletionRate.toFixed(1)}%</div>
      <div style={{width:600}}>
        <Line data={{ labels, datasets:[{ label:'Bookings', data }] }} />
      </div>
      <button onClick={()=>handleExport('bookings')}>Export Bookings CSV</
button>
      <button onClick={()=>handleExport('finances')}>Export Finances CSV</
button>
    </div>
  );
}
JS

cat > frontend/src/pages/Automation.jsx <<'JS'
```

```
import React, { useEffect, useState } from 'react';
import { api } from '../api';

export default function Automation(){
  const [rules, setRules] = useState([]);
  const [form, setForm] = useState({name:'', trigger:'', action:''});
  useEffect(()=>{ api.get('/
automations').then(r=>setRules(r.data)).catch(console.error) },[]);
  const create = async () => { const res = await api.post('/automations',
form); setRules(prev=>[...prev, res.data]); setForm({name:'', trigger:'',
action:''}); }
  const remove = async (id) => { await api.delete(`/automations/${id}`);
setRules(prev=>prev.filter(r=>r.id!==id)); }
  return (
    <div>
      <h2>Automation & Alerts</h2>
      <p>Define simple rules to schedule reminders and notifications.</p>
      <div>
        <input placeholder="Name" value={form.name}
onChange={e=>setForm({...form,name:e.target.value})} />
        <input placeholder="Trigger (e.g. booking_due)" value={form.trigger}
onChange={e=>setForm({...form,trigger:e.target.value})} />
        <input placeholder="Action (e.g. email_reminder)" value={form.action}
onChange={e=>setForm({...form,action:e.target.value})} />
        <button onClick={create}>Create Rule</button>
      </div>
      <ul>
        {rules.map(r=> (
          <li key={r.id}>{r.name} — {r.trigger} — {r.action} <button
onClick={()=>remove(r.id)}>Delete</button></li>
        ))}
      </ul>
    </div>
  );
}
JS

cat > frontend/src/style.css <<'CSS'
body { font-family: Inter, Arial, sans-serif; }
input { margin: 6px; padding:6px }
button { margin:6px }
CSS

# Install frontend deps
(cd frontend && npm install react react-dom react-router-dom axios chart.js
react-chartjs-2 json2csv jspdf vite --save)

echo "
Setup complete.
Files created under ./backend and ./frontend
Run backend: node backend/server.js
```

```
Run frontend: npm --prefix frontend run dev
"
```