# Solo Accounting App — Master Checklist (Learning → MVP → v1)

This checklist mirrors the Markdown version exactly, formatted for Google Docs or Microsoft Word.

# Solo Accounting App — Master Checklist (Learning → MVP → v1)

This checklist is organized to be \*\*followed in order\*\*. Each item is intentionally small and shippable.

Mark items `[x]` as you complete them. You can paste this whole file into a new chat to give context.

## 0) Orientation & Working Style

* ☐ Define scope level (Tiny MVP / MVP+ / Ambitious v1)
* ☐ Create a simple time budget (hrs/week) + target end date
* ☐ Decide source of truth (this checklist in Git + one README)
* ☐ Decide issue tracker (GitHub Issues/Projects) and definition of done
* ☐ Set up a weekly cadence (plan → build → demo → retro)

## 1) Local Environment & Tooling

* ☐ Install VS Code (extensions: Python, Pylance, Docker, GitHub Copilot or Cursor)
* ☐ Install Python (pyenv or system) and pipx / uv
* ☐ Install Git and create GitHub account
* ☐ Set up a private repo `accounting-backend`
* ☐ Create a `.gitignore` for Python, node, env files
* ☐ Install Postman/Insomnia for API testing

## 2) Python Foundations (hands-on)

* ☐ Write a script that reads a CSV and prints filtered rows
* ☐ Write a script with functions, exceptions, and type hints
* ☐ Package + virtualenv basics (pip/uv, requirements/pyproject)
* ☐ Unit tests with `pytest` (one passing test, one failing test)

## 3) SQL & PostgreSQL Basics

* ☐ Install PostgreSQL locally (user + password)
* ☐ Create a database `acct\_app\_dev`
* ☐ Practice: `CREATE TABLE`, `INSERT`, `SELECT`, `JOIN`, `INDEX`
* ☐ Backup/restore with `pg\_dump` and `psql`

## 4) FastAPI Basics

* ☐ Create a `hello world` FastAPI app with `/health`
* ☐ Run with `uvicorn` and hot reload
* ☐ Add request/response models with Pydantic
* ☐ Add error handlers (404/400/500); return JSON errors

## 5) Project Scaffolding

* ☐ Create folders: `app/api`, `app/models`, `app/schemas`, `app/services`, `app/db`, `app/core`
* ☐ Add settings via `pydantic\_settings` (`.env` file)
* ☐ Setup logging formatter with request IDs
* ☐ Add `pre-commit` (black/isort/ruff) for formatting/lint

## 6) DB Layer & Migrations

* ☐ Add SQLAlchemy and session management
* ☐ Install & configure Alembic
* ☐ Create migration `0001\_init` with `clients` table
* ☐ Write seed script (use `faker`) to insert demo clients

## 7) Core Domain Models (Schemas + Tables)

* ☐ Clients (id, name, email, phone, company, notes, created\_at)
* ☐ Contacts (client\_id, role, email, phone)
* ☐ Engagements (client\_id, type, status, start/end dates)
* ☐ Tasks (engagement\_id, title, due\_at, status, assignee\_id, priority)
* ☐ Users (id, email, password\_hash, role)
* ☐ Invoices (client\_id, amount, status, due\_at, paid\_at, stripe\_id)
* ☐ Files (owner\_type/id, path, size, content\_type, created\_at)
* ☐ Notifications (type, to, subject, body, status, retries, last\_error)

## 8) CRUD Endpoints (MVP)

* ☐ `/clients` (list, filter, create, retrieve, update, delete)
* ☐ `/contacts` (CRUD)
* ☐ `/engagements` (CRUD)
* ☐ `/tasks` (CRUD + list by client/assignee + status transitions)
* ☐ Pagination, sorting, basic search for each list endpoint
* ☐ Basic input validation + error messages

## 9) Auth & RBAC

* ☐ Add password hashing (argon2/bcrypt)
* ☐ JWT login + refresh
* ☐ Role enum: `admin`, `staff`, `client`
* ☐ Route protection decorators/dependencies
* ☐ Audit log: record `user\_id`, action, target, timestamp

## 10) Background Jobs & Scheduling

* ☐ Add APScheduler (or RQ/Celery later)
* ☐ Nightly job: check overdue tasks and queue reminders
* ☐ Store job runs and outcomes in `jobs` table/logs

## 11) Email & SMS (Notifications)

* ☐ Choose providers (SendGrid for email, Twilio for SMS)
* ☐ Secret management: store API keys in env
* ☐ Implement `NotificationService` with retries & dead-letter
* ☐ Templates: Jinja2 email templates for reminders
* ☐ Test “sandbox” emails to yourself

## 12) Calendar Integration (Google first)

* ☐ Create Google Cloud project & OAuth credentials
* ☐ Implement OAuth flow and token storage
* ☐ One-way sync: create calendar events from tasks
* ☐ Webhook endpoint to receive updates (push notifications) — optional at MVP
* ☐ Manual “resync” endpoint

## 13) File Storage & Client Portal Prep

* ☐ Pick S3-compatible storage (AWS S3 / Wasabi / Backblaze)
* ☐ Create bucket and IAM keys; set lifecycle rules
* ☐ Upload endpoint with presigned URLs
* ☐ File metadata table + access control checks
* ☐ Virus scan stub (mark as TODO for later if needed)

## 14) E-Signatures (Engagement Letters)

* ☐ Pick provider (DocuSign / Dropbox Sign)
* ☐ Create a simple envelope/template
* ☐ Send for signature via API
* ☐ Webhook: mark signed → advance workflow step

## 15) Invoicing & Payments (Stripe)

* ☐ Setup Stripe account + test keys
* ☐ Create products/prices or use ad-hoc amounts
* ☐ Checkout session endpoint
* ☐ Webhook: record payment success/failure
* ☐ Generate invoice PDF (WeasyPrint/ReportLab) and email

## 16) Minimal Client Portal (Next.js)

* ☐ Bootstrap Next.js app (TypeScript)
* ☐ Auth via your backend (HTTP-only cookies or token exchange)
* ☐ Pages: Login, My Tasks, Uploads, Invoices/Pay, Messages (MVP)
* ☐ Use your existing REST API; no overengineering
* ☐ Simple design system (shadcn/ui or minimal CSS)

## 17) Smart/AI Features (choose 1–2 first)

* ☐ AI Proposal Builder: prompt on client + engagement + templates
* ☐ AI Transcription: upload audio → transcript → auto-create tasks
* ☐ OCR + Search: extract PDF text; search endpoint
* ☐ RAG for internal SOPs: store chunks; answer questions in staff UI

## 18) Observability & Quality

* ☐ Structured logging (JSON) with request IDs
* ☐ Centralized error tracking (Sentry or similar)
* ☐ Health checks + uptime monitor (cron ping or external service)
* ☐ API rate limiting (middleware) and sensible timeouts
* ☐ Basic load test (Locust/k6) on a couple endpoints

## 19) Security & Compliance Basics

* ☐ Enforce HTTPS; secure cookies; CORS tightened
* ☐ Input validation on every boundary (API, webhooks, uploads)
* ☐ Least-privilege DB user; rotate secrets
* ☐ Backups: nightly DB + verify restore
* ☐ PII review: mask/scrub in logs; access controls for files
* ☐ Terms/Privacy stubs (for later legal review)
* ☐ SOC2/IRS-4557 mapping: note which controls you already meet

## 20) Testing Strategy

* ☐ Unit tests for services and utils
* ☐ API tests (pytest + httpx) for happy/edge cases
* ☐ Integration tests with a test DB
* ☐ Minimal E2E (scripted Postman collection) for the main user journey

## 21) Docker & Local Parity

* ☐ Dockerfile for backend
* ☐ docker-compose for backend + Postgres + workers
* ☐ Run migrations on container start
* ☐ Dev vs prod configs

## 22) Deployments

* ☐ Pick host (Render/Railway/Fly.io)
* ☐ Provision managed Postgres
* ☐ Set env vars & secrets in host dashboard
* ☐ Add health checks and start command
* ☐ Domain + TLS (if needed)

## 23) CI/CD

* ☐ GitHub Actions: run tests on PR
* ☐ Build and push container image
* ☐ Auto-deploy `main` on tag or `release/\*` branch
* ☐ Keep `.env.example` in repo

## 24) Data Safety & Backups

* ☐ Automatic daily DB backups; retain 7/30/90
* ☐ S3 bucket versioning or object locking (if available)
* ☐ Disaster recovery runbook: restore in a fresh environment

## 25) Performance & Indexing

* ☐ Add indexes for frequent filters (client\_id, status, due\_at)
* ☐ Measure slow queries; add EXPLAIN plan to docs
* ☐ Cache hot reads (simple in-process or Redis later)

## 26) Multi-tenant & Roles (if needed)

* ☐ Tenant model (org\_id on all tables) or schema-per-tenant
* ☐ Enforce tenant in queries via middleware
* ☐ Admin vs staff vs client capabilities documented

## 27) Documentation & Onboarding

* ☐ `README` with setup, run, test, deploy
* ☐ `ARCHITECTURE.md` with diagrams (DB ERD + component overview)
* ☐ `OPERATIONS.md` (secrets, backups, rotations, deployments)
* ☐ `SECURITY.md` (threat model & mitigations)
* ☐ Changelog for releases

## 28) Pre-Launch QA & MVP Hardening

* ☐ Walk the main flows end-to-end (staff + client)
* ☐ Verify email/SMS from sandbox to live
* ☐ Verify Stripe live small transaction
* ☐ Verify OAuth consent screen + scopes
* ☐ Verify file upload/download permissions
* ☐ Verify error logging & alerts fire

## 29) Post-Launch

* ☐ Collect feedback; triage into issues
* ☐ Add metrics dashboard for daily active clients, tasks, payments
* ☐ Plan next 4–6 weeks of improvements

## Appendix: Useful Starter Commands

* ☐ `uv pip compile` / `pip-compile` to lock deps (or just `pip install -r requirements.txt`)
* ☐ `alembic init` → `alembic revision --autogenerate -m "init"` → `alembic upgrade head`
* ☐ `uvicorn app.main:app --reload`
* ☐ `pytest -q`