perplexity

Enhanced Sprint Plan with Detailed Developer Instructions

This revision deepens the guidance previously supplied by spelling out **day-by-day activities, acceptance criteria, deliverables, and hand-off checkpoints** for each of the four developers across all three sprints. All timings assume an 8-hour workday.

Sprint 1 – Foundation (Week 1)

Day	Dev 1 – Backend Lead	Dev 2 – Data Engineer	Dev 3 – Algorithm Dev	Dev 4 – Frontend / Integr
Mon (D1)	Create project repo, set up virtual-env, commit base `pyproject.toml`; stub `qloo_client.py` class; request Qloo API key and store in `.env`[1]	Fork repo, add `data/grocery_catalog.csv`, draft schema for `associations.db`; install pandas, sqlite3	Draft `AssociationEngine` interface; outline unit-test matrix in `tests/test_association_engine.py`	Scaffold Streamlit app (`streamlit init` shell with empty pages
Tue (D2)	Implement GET `/recommendations/products` wrapper with retry & exponential back-off; write happy-path & timeout unit tests	Build ETL script to import catalog into SQLite; publish Makefile target `make load_catalog`	Write parsing helpers (`parse_associations`) incl. Pydantic models for response validation	Add sidebar brand, configure Tailwind CS for UI components
Wed (D3)	Finish POST `/insights/preferences` wrapper; benchmark latency (<2 s); push coverage badge	Develop batch job to call API for 1000 items/hr; use `tqdm` progress bar; persist to SQLite	Implement `process_grocery_catalog()` calling Dev 1 wrappers; unit tests ≥80%	Design "Association Network" placeholde event bus for real-time updates
Thu (D4)	Harden API module: circuit-breaker, 429 handling, ENV-switch for sandbox/production	Add incremental update logic using `last_updated` column; write DDL in `database/schema.sql`	Compute confidence scores; create top-5 association view; expose via `associations.to_parquet()`	Hook Parquet feed; render demo heatmap profile render time (<1 s)
Fri (D5)	Code review & merge Dev 1+3 PRs; publish package to internal PyPI	Profiling & optimization (indexing, WAL, PRAGMA tuning) – queries <100 ms for 10 k rows[2]	Pair with Dev 4 to validate algorithm JSON contract	Implement `layout_optimizer.py::generate_sh` returning ≥200 placements[1][2]
Sat (D6)	End-to-end smoke test: fetch 10 SKUs → DB → association JSON	Write nightly cron (`schedule` lib) for batch associations → push to branch `feature/cron`	Draft docstring examples; ensure mypy passes	Visual smoke test inside Streamlit; fix lint
Sun (D7)	Buffer / contingency; harden docs (`docs/api.md`)	Buffer / contingency	Buffer / contingency	Buffer / contingency

Sprint 1 Acceptance Criteria

1. `qloo_client.py` returns valid JSON for sample SKUs in <2 s round-trip[1].
2. `associations.db` contains ≥10 000 product-pair rows, <100 ms avg read time[2].
3. `layout_optimizer.generate_shelf_placements()` outputs ≥200 placement dicts with schema `{primary_product, companion_product, confidence, shelf_distance}`.
4. All modules achieve ≥80% unit-test coverage and pass on CI pipeline.

Sprint 2 – Core Features (Week 2)

Day	Dev 1 – Backend Lead	Dev 2 – Data Engineer	Dev 3 – Algorithm Dev	Dev 4 – Frontend / Integration
Mon (D8)	Branch `feature/combo_generator`; design dataclass `Combo`	Draft weekly KPI table design for reports; set up Jinja2 templates	Extend optimizer for **section-level** placement (`optimize_store_sections`)	Flesh out "Shelf Placements" page: DataTable + sortable confidence column
Tue (D9)	Implement high-confidence filter (≥0.8) and `generate_weekly_combos()`; unit tests	Build `WeeklyReport` model → YAML; accept CLI `--week YYYY-MM-DD`	Implement heuristic: crowding penalty + adjacency reward; parameterize via `config.yml`	Add Plotly Sankey diagram for product flow; anchor to section IDs
Wed (D10)	Integrate price API stub to suggest discount %; ensure configurable fallback	Aggregation SQL views for: new associations vs prior week; expose via Pandas	Refactor scoring into `algorithms/scoring.py`; publish ROC metrics notebook	Build "Combo Offers" UI with card layout; add CTA "Download JSON"

Day	Dev 1 – Backend Lead	Dev 2 – Data Engineer	Dev 3 – Algorithm Dev	Dev 4 – Frontend / Integration
Thu (D11)	Dockerize service `association_api` (FastAPI) to expose `/combos` endpoint	Schedule Airflow DAG (`weekly_report`) triggered Fridays 02:00; load to `reports/`	Benchmark optimizer runtime <180 s for 10 k SKUs; profile hotspots	Integrate Dag status feed into Streamlit (`st_autorefresh`)
Fri (D12)	Publish OpenAPI schema & Swagger docs; PR review	Unit tests for DAG idempotency; add Slack webhook on success/fail	Document tunable hyper-params; write Sphinx page	Optimize UI bundle size (<1 MB) by code-splitting; Lighthouse score ≥90
Sat (D13)	End-to-end test: `/combos` returns ≥20 offers with JSON schema	Generate sample HTML report; attach inline heatmaps via base64	Validate optimizer output with new offers; cross-check anomalies	Add responsive design fixes; QA on mobile viewport
Sun (D14)	Buffer / bug fixes	Buffer / bug fixes	Buffer / bug fixes	Buffer / bug fixes

Sprint 2 Acceptance Criteria

1. REST endpoint `/combos` delivers ≥20 weekly combos with ≥80% Qloo confidence[1].
2. Airflow DAG runs automatically and writes HTML + assets in <5 min[1] [2].
3. `optimize_store_sections()` optimizes 8 predefined sections with crowding penalty ≤5%.
4. Streamlit pages (“Associations”, “Placements”, “Combos”, “Reports”) load in <1 s (95th pct).

Sprint 3 – Interface & Polish (Week 3)

Day	Dev 1 – Backend Lead	Dev 2 – Data Engineer	Dev 3 – Algorithm Dev	Dev 4 – Frontend / Integration
Mon (D15)	Integrate modules via service mesh; write `docker-compose.yml`	Design OLTP schema for future POS ingestion; add migration scripts (Alembic)	Draft “association drift” detection notebook for future ML	Build dashboard homepage: KPI cards pulling from `/metrics`
Tue (D16)	Set up GitHub Actions: lint-test-build-deploy; add cache	Bulk load sample POS CSV; stress-test DB (10 k products, 1 M tx)	Author `docs/architecture.md` diagrams (Mermaid)	Implement auth layer (basic HTTP password) for dashboard
Wed (D17)	JMeter perf test: 50 RPS, target API p95 <2 s; tune gunicorn workers	Create read replicas & benchmarks; finalize indices	Polish docstrings; run `pydocstyle`; generate API docs	Add interactive store map (Plotly) with placement overlays
Thu (D18)	Write 100+ integration tests; branch protection rules	Add DB backup script + cron; test restore in staging	Package CLI (`smo-cli`) for headless invocation; upload to PyPI test	UI polish: dark/light toggle, accessibility audit
Fri (D19)	Orchestrate deployment script `deployment/setup.sh` for fresh Ubuntu; 30-min install target[2]	Tune connection pooling; ensure query p95 <300 ms	Prepare CHANGELOG, semantic version tags (`v1.0.0-rc1`)	Cross-browser QA; fix edge cases in Safari
Sat (D20)	Final load test & harden security headers; OWASP ZAP scan 0 highs	Snapshot production DB; stage migration docs	Conduct dry-run hand-over demo; capture video	Add README badges (build, coverage, license)
Sun (D21)	Sprint review demo lead; ensure 99% uptime in staging since midnight	Present DB metrics; hand-off run-books	Present algorithm notebook & future roadmap	Live demo of full dashboard; gather feedback

Sprint 3 Acceptance Criteria

1. **Streamlit dashboard** offers 5 sections, authenticated, mobile-responsive, p95 load <1 s[1].
2. System withstands 50 RPS for 30 min with uptime ≥99%, memory <1 GB/container[2].
3. Fresh-host deployment completes in ≤30 min via `deployment/setup.sh`, no manual edits.
4. Full documentation (architecture, API, ops, user guide) published and linked from README.

Cross-Team Standards

Area	Standard
Branch naming	`feature/<slug>`, `bugfix/<slug>`, `hotfix/<slug>`
Commit style	Conventional Commits (`feat:`, `fix:`, `docs:` …)

Area	Standard
Pull Requests	1 reviewer min; include screenshot/benchmark where relevant
Testing	pytest + coverage; failing tests block merge
CI	Lint → Unit tests → Build image → Push to registry → Deploy to staging
Definition of Done	Code, tests, docs, review, performance budget, acceptance tests pass

Sprint Ceremonies & Artefacts

Ceremony	Time-box	Artefact
Sprint Planning	2 h Monday 09:00	JIRA sprint board with story points and owner
Daily Stand-up	15 m 09:00	Slack summary thread (“Yesterday / Today / Blockers”)
Mid-sprint Demo	30 m Wednesday	Loom video or live Zoom showing progress
Sprint Review	1 h Sunday	Working software demo, KPI dashboard snapshot
Retrospective	1 h Sunday	“Start / Stop / Continue” Miro board; action items assigned

By following these **granular, outcome-oriented instructions**, each developer knows exactly **what to code each day, how to validate it, and which measurable criteria define success**—ensuring visible, high-quality deliverables at the close of every sprint.

⁂

1. Qloo-Powered-Supermarket-Layout-Optimizer-Implementation-Plan.pdf
2. Supermarket-Product-Association-Layout-Optimization-Requirements-Document.pdf