63LABS

Restauranteur Learn & Build

SECTION 04 OF 16

Team Operations and Procedures

Procedures, training, scheduling, task ownership, risk, continuity and operating reviews.



SEPARATE TOPIC PDF / FULL RESTAURANT EDITION PRESERVED

Read this section

Two page-number systems

The section contents below use PDF page positions in this file. Original lesson footers keep their page numbers from the complete 619-page Restaurant Edition. "Master p/pp" still refers to the original 5,410-page Learn & Build source.

Same lessons, smaller files

Content has been separated, not rewritten. Nine shared-boundary pages were divided between their topics so every lesson keeps its ending. The complete Restaurant Edition and original master remain unchanged.

Start with the shared guide

The original scope guide is included below. Project sections also include the shared build starter and spreadsheet setup. These are learning and build specifications, not finished apps or certified operating instructions.

Moving between sections

Local contents and bookmarks open pages in this PDF. Cross-section links use the companion filenames and may require all files in one folder and a supporting PDF reader. If a link does not open, use the numbered section list and the printed edition-page label. Section 01 retains the original full-edition contents and source index.

The 16-section collection

01 Start Here and Source Index

02 Business Planning and Money

03 Stock Suppliers and Food Records

04 Team Operations and Procedures

05 Facilities Equipment and Resources

06 Catering Guests and Delivery

07 Marketing Publishing and Websites

08 Restaurant Products and Services

09 Careers and Team Learning

10 AI Tools and Task Guides

11 Operations Methods and Data

12 Evidence and Reasoning Cases

13 Integrated Restaurant Projects

14 Guest Clubs and Event Ideas

15 Supplier and Product Ideas

16 Concepts and Quality Checks

Contents for this PDF

Numbers on the right are PDF page positions in this section. Retained lesson footers show their complete-edition page labels.

How to use this edition (shared guide) 4

Shared build starter (shared guide) 5

Spreadsheet and document setup (shared guide) 6

Procedures, people and delivery 7

P012 Restaurant launch and delivery projects 8

P014 Opening, closing and shift-handoff procedures 10

P015 Restaurant role training and sign-off 12

P035 Catering consultations and venue appointments 14

P040 Restaurant-service time and project billing 16

P044 Restaurant operational risk register 18

P045 Restaurant service continuity plan 20

P046 Restaurant meetings and accountable actions 22

P051 Restaurant manager weekly workload and capacity planner 24

P052 Opening, closing, and manager-admin routine occurrence log 27

P053 Recurring service-practice and manager habit tracker 30

P054 Restaurant improvement goal and milestone dashboard 32

P055 Owner or manager operating review workspace 35

P062 Private post-service learning reflection and optional team-appreciation notes 38

P074 Shared restaurant-team, vehicle, and event-resource calendar 40

P075 Restaurant duty rotation and task-load coordination board 42

How to use this edition

This is a restaurant-focused curation of the actual 5,410-page 63Labs Learn & Build combined master. It keeps useful original lessons, projects, worked reasoning and tool guidance, then adds clearly labeled restaurant applications. The original file remains unchanged.

A reference library rather than a course

Start with the problem you are solving. Operators can use the money, stock, supplier, team and guest workflows; restaurant-industry builders can use the service, software, content and commercial project material. You do not need to implement every project or buy every named tool.

Broad restaurant use, not just daily operations

The edition retains all 50 original business projects and 43 adaptable planning, facilities, food-record, event and learning projects. Optional supplier, catering, career, venue-event and creative applications are identified as such. Private health, medication, sleep, fitness, mood and veterinary-care projects are not converted into employee-management tools.

What was changed

Repeated setup instructions and repeated practice layouts are consolidated. Restaurant application paragraphs are new adaptations. Source lessons and project requirements are identified by their original master page numbers. Necessary source corrections are labeled rather than silently presented as original text.

What you receive

A learning and build-specification PDF. It does not include generated apps, connected POS systems, live forms, installed automations, completed spreadsheets, legal policies, or a restaurant-specific food-safety program. The projects explain what to build and how to test it.

Source references

Master p or pp always means the physical page position in 63Labs_Combined_Ready.pdf, not a page in this edition. P001-style labels preserve the original project IDs. The contents and PDF bookmarks navigate this edition; the final index traces its source material.

Keep consequential decisions reviewed

Use fictional data while building. Keep private guest, staff and payment data out of shared examples. Use approved restaurant procedures and current authoritative guidance for food safety, employment, taxes, accessibility, privacy and other regulated matters. This edition does not certify compliance or safe equipment work.

Shared build starter

Use this once per build, followed by either the spreadsheet setup or document setup, then the chosen project specification. These consolidated instructions preserve the common safeguards from the source rather than repeating them in every project.

Copy this starter

Act as a patient build partner. Ask the project intake questions in one message, then wait for my answers. I may answer not sure. Separate my facts, goals, assumptions and research still needed. Never invent business facts, sources, permissions, credentials, customers, results or regulatory requirements. Use fictional sample records. Produce the complete requested artifact, not merely an outline. Explain how to copy, set up, use, test and revise it. Do not claim you created or saved a file unless you actually did. If long, stop at a section boundary and resume when I say continue without omissions. End with a requirement-by-requirement checklist. Do not connect services, publish, send messages, spend money or change live records as a side effect of drafting this build.

Add my restaurant context

My restaurant or industry role: [role]. Intended user: [person or role]. Exact task: [task]. Scope and exclusions: [limits]. Approved sources: [documents and dates]. Tools and access available: [tools]. Currency, locale and units: [values]. Owner and reviewer: [roles]. Deadline or work budget: [limit]. Ask about missing information rather than assuming it.

If a response or formula fails

Continue from the exact point where you stopped; keep the same structure, tab names, column order and references. For a formula issue, inspect the actual formula, cell, expected result, source values, locale, dates, units and copied ranges. Explain the proposed correction and rerun the original test. A polished answer is not proof of correctness.

Spreadsheet and document setup

Spreadsheet setup

For Google Sheets, confirm locale and units. The source examples use English functions, comma separators, real dates displayed YYYY-MM-DD, headers in row 1 and data rows 2:1000. Preserve named tabs and column order. Supply actual tab-separated headers, fictional sample rows and exact paste cells. Leave computed sample cells empty. Give each formula with its exact cell, explanation, fill range and expected result. Keep unused rows blank and distinguish missing inputs from zero. Specify validation, dropdowns, formats, filters, frozen headers and formula protection. Keep raw data tables unmerged. Do not add scripts, paid add-ons or external connections by default. Explain how every related range extends beyond row 1000.

Document setup

Create copy-ready pages with clear headings, useful tables only where appropriate, fill-in fields and a completed fictional example. Keep the reusable blank version separate from the example. Supply the actual wording and templates requested, not instructions to write them later. Name owners, actions, dates and review points. Use approved policy content; for changing rules, cite current primary sources only when verified, otherwise list a precise verification task.

First use

Create a safe copy. Enter the source example and compare every expected value. Test blank inputs, explicit zero, duplicate IDs, refunds/reversals, period boundaries and a changed value. For dates and percentages, inspect the underlying values as well as the display. Save evidence before replacing fictional records with authorized live information.

Limits of the supplied specifications

The arithmetic examples have been reviewed, but this PDF is not an executed application test. Generated formulas, data validation, access controls, links, integrations and regulatory treatment still need testing in the actual implementation. Editorial clarifications identify known source gaps, not a claim that all possible edge cases are covered.

Procedures, people and delivery

Use the projects below independently. P numbers are the original master project IDs, retained for traceability. The short source lessons and worked reasoning are followed by project-specific build specifications.

P012 Restaurant launch and delivery projects

P014 Opening, closing and shift-handoff procedures

P015 Restaurant role training and sign-off

P035 Catering consultations and venue appointments

P040 Restaurant-service time and project billing

P044 Restaurant operational risk register

P045 Restaurant service continuity plan

P046 Restaurant meetings and accountable actions

P051 Restaurant manager weekly workload and capacity planner

P052 Opening, closing, and manager-admin routine occurrence log

P053 Recurring service-practice and manager habit tracker

P054 Restaurant improvement goal and milestone dashboard

P055 Owner or manager operating review workspace

P062 Private post-service learning reflection and optional team-appreciation notes

P074 Shared restaurant-team, vehicle, and event-resource calendar

P075 Restaurant duty rotation and task-load coordination board

P012 Restaurant launch and delivery projects

Procedures, people and delivery

Source project: Project Management Project. Master pp 1023, 1163-1164, 2972. Restaurant application below is an editorial adaptation.

Restaurant application

Plan a new menu, opening, refurbishment, catered event or website delivery with tasks, owners, dependencies and evidence of completion.

Source based lesson

Organize projects, tasks, milestones, dependencies, and weekly workload with a clear definition of completion.

Gather and build

Project ID, task owner, deadline, dependency, completion evidence. Break one outcome into tasks; identify blocked work; define what finished means before starting.

Check before relying on it

A task cannot be complete solely because its due date passed. Run a weekly review of blocked tasks and revise the next actions.

Source based practice with fictional inputs

Drafting is complete. Approval is pending. Export depends on approval.

Worked result: Export remains blocked until the approval event is recorded. Use completion evidence rather than dates to advance state.

Try a changed condition: Approval arrives with a new requirement to include a second format.

Check your answer: Record the changed requirement and finish both required exports before completion.

Restaurant boundary

Dates do not prove progress. Permit, contractor and supplier commitments must be confirmed; the project is not construction or food-safety instruction.

P012 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

1. What projects and outcomes are in scope?
2. Who owns tasks and how many hours are available each week?
3. What fixed deadlines and dependencies exist?
4. What counts as Done?
5. Do you prefer a list, status board, or timeline view?

Structure

Projects A:F = ProjectID, Project, Owner, StartDate, TargetDate, Status. Tasks A:L = TaskID, ProjectID, Task, Owner, StartDate, DueDate, EstimatedHours, Status, DependencyID, CompletedDate, NextAction, DueFlag. Status values: Backlog/Ready/Doing/Blocked/Done/Cancelled. Add Milestones and Review tabs.

Build Requirements

Tasks L2: =IF(A2="","",IF(OR(H2="Done",H2="Cancelled"),H2,IF(F2="","No date",IF(F2<TODAY(),"Overdue",IF(F2=TODAY(),"Today","Upcoming"))))) Validate StartDate<=DueDate when both exist. Flag unknown ProjectID, missing owner, duplicate TaskID, Done without CompletedDate, and a dependency pointing to itself. For the base system allow one DependencyID per task; a dependency is met only when the referenced task is Done. Do not automatically mark a task Done because its date passed. Detect missing dependency IDs; explain that multi-step cycles require a manual chain review.

Milestones: MilestoneID, ProjectID, Deliverable, DueDate, AcceptanceEvidence, Status. Project progress is Done task count divided by non-Cancelled task count, with no tasks shown as No tasks. Do not label this effort-weighted progress. Give COUNTIFS formulas and show open/blocked task counts. Weekly workload is the sum of EstimatedHours on open tasks allocated to the selected week; add a WeekStart input column if tasks span weeks and explain splitting estimates without duplicate TaskIDs.

Create filtered views for My next actions, Overdue, Blocked, and This week. A status board can be a filtered view of one source table. Provide a weekly review agenda: close evidence-backed work, resolve blockers, rebalance estimates, choose next milestones. Test four tasks: two Done, one Doing, one Cancelled. Completion is 2/3, not 2/4; cancelled work is excluded.

Acceptance Check

The four-task example returns 2/3; missing dependencies, missing due dates, and overdue open work are flagged. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1165. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Team views: Create owner-specific views for [names] while preserving one master Tasks table.

Capacity: Add weekly planned hours and capacity for each owner, with an over-capacity flag and no automatic rescheduling.

Recurring work: Add a template for recurring tasks. Each occurrence must receive a new TaskID and preserve prior completion records.

Progress looks too high. Check whether cancelled tasks or empty rows entered the denominator.

Dependency never clears. Match the exact permanent TaskID and inspect its status.

Hours repeat. Use one weekly allocation per task occurrence; do not count both parent and child estimates.

Due date becomes text. Enter a real date and reapply the date format.

P014 Opening, closing and shift-handoff procedures

Procedures, people and delivery

Source project: Procedures Project. Master pp 1025, 1171-1172, 2996. Restaurant application below is an editorial adaptation.

Restaurant application

Build the restaurant operations toolkit from actual manager-approved procedures, one process at a time.

Source based lesson

Turn one repeatable business process into a usable standard operating procedure with decision points and a test run.

Gather and build

Process trigger, inputs, owner, steps, exceptions, output. Observe the task; write numbered actions and decision points; ask someone else to follow the procedure.

Check before relying on it

A new reader can complete a dry run without asking what each step means. Revise the first unclear instruction and add a version date.

Source based practice with fictional inputs

A draft procedure says check the file, approve it, and send it.

Worked result: A usable procedure must name the authoritative file, check criteria, approver, and recipient. Specify file identity and version, the check, and the approval evidence.

Try a changed condition: The named approver is absent.

Check your answer: Use a defined backup or escalation route; do not invent approval.

Restaurant boundary

Do not invent cleaning chemistry, food-safety procedures, equipment instructions or regulatory rules. Unverified procedures stay Draft. Preserve pilot, approval, version and evidence logs.

P014 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

1. What exact process and trigger are in scope?
2. Who performs and approves it?
3. What inputs, permissions, and tools are required?
4. What is the current sequence, including exceptions?
5. How do you verify successful completion?

Structure

Create an SOP cover/control block, purpose and scope, prerequisites, numbered procedure, decision table, quality checklist, exception path, evidence log, and version history.

Build Requirements

Use a control block with SOP ID, title, version, process owner, approver, effective date, review date, and status Draft/Approved/Retired. Keep unverified procedures in Draft. Define the start trigger and exact end state. List required inputs, roles, access, tools, and exclusions.

Write each step with an action verb, responsible role, input, expected result, and check before proceeding. Break ambiguous steps such as "process the request" into observable actions. Include screenshot placeholders only where they would clarify a known interface; do not invent interface labels. A decision table must specify condition, action, owner, and escalation for missing input, failed checks, and an unavailable approver.

Create a short execution checklist and an evidence log: RunID, Date, Operator, InputRef, Result, Exception, Reviewer. Separate troubleshooting from the normal path. For irreversible or sensitive steps include the actual authorization gate supplied by me. Do not fabricate safety procedures or regulatory requirements; mark source/manual verification tasks.

Provide a fictional worked example for processing a routine customer address-update request: verify the authorized requester according to the supplied policy, check the new address, update the permitted record, verify it, and record completion. Use fictional details. Show a missing-authorization exception that stops the update and routes for review.

Finish with a pilot plan: a new operator follows the draft, records unclear steps and elapsed time, the owner revises it, and the approver signs off. Add a change-request template and archive instructions so outdated copies are identifiable.

Acceptance Check

A new operator can complete the normal path and identify when to stop; version, owner, approval, and evidence fields are present. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1173. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Simplify: Rewrite the procedure for a new starter using [reading level], preserving controls and decision points.

Team version: Adapt roles and escalation to [team structure], with one accountable owner per step.

Audit evidence: Add evidence fields required by [our supplied policy], marking any requirement not supported by that policy.

Steps are vague. Ask for observable actions and an expected result after every step.

Too many exceptions. Keep the main path short and move conditional branches into the decision table.

Versions conflict. Choose the approved source, retire superseded copies, and record the change.

Test run fails. Capture where the operator got stuck and revise that exact step before approval.

P015 Restaurant role training and sign-off

Procedures, people and delivery

Source project: Team Training Project. Master pp 1026, 1175-1176, 3008. Restaurant application below is an editorial adaptation.

Restaurant application

Adapt to a host, server, shift lead or other actual role with supervised practice from approved procedures.

Source based lesson

Build a role-specific training manual with practice activities, competency checks, and a clear first-month learning path.

Gather and build

Role duties, prerequisites, approved procedures, mentor, assessment. Sequence observation, guided practice, independent demonstration, and review; define evidence for each skill.

Check before relying on it

Reading a manual alone does not prove practical competence. Create one assessment with observable criteria and a reviewer sign-off.

Source based practice with fictional inputs

Four learners attend. Two complete independently, one completes with coaching, one does not attempt.

Worked result: Record four attendees, two independent demonstrations, one assisted attempt, and one unattempted assessment. Separate attendance from observed performance and schedule relevant follow-up.

Try a changed condition: The assisted learner later completes independently under the same criteria.

Check your answer: Update that learner's demonstrated status with the date and evidence; retain the earlier attempt.

Restaurant boundary

Reading is not competence. Preserve critical-step blocks and reassessment history; do not invent employment or safety policies.

P015 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

1. Which role and experience level is this for?
2. Which tasks must the person perform independently?
3. What approved SOPs and policies may be used?
4. What access and training time are available?
5. Who evaluates competency and what evidence is required?

Structure

Create a welcome and role overview, source register, access checklist, first-day/first-week/30-day schedule, six training modules, practice tasks, assessment rubrics, sign-off record, and manager review form.

Build Requirements

Ground all policy content in the materials I provide. Quote only short necessary excerpts and otherwise summarize with a source/section reference. If a required policy is missing, mark it "pending approved source"; do not invent employment terms, safety procedures, benefits, disciplinary rules, or legal obligations.

Build six modules based on the role's actual tasks. Each module needs objective, prerequisite, estimated learning time, trainer, demonstration steps, learner practice, common errors, and a measurable pass criterion. At least one module must cover finding help and escalating an uncertain task. A read receipt is not a competency sign-off.

Create a practical assessment with three scenarios and a rubric scored 0=not yet demonstrated, 1=with support, 2=independently demonstrated. Mark critical steps separately; the owner defines which must pass before independent work. Provide sample model responses as training examples, not universal policy. Record attempts, feedback, retraining, and reassessment dates without overwriting history.

The 30-day schedule must fit the available training hours, include supervised practice and breaks, and state what is deferred. Include a glossary and a source register with document owner, version, location, and review date. Provide a learner question log and manager check-in prompts.

Example: a support associate practices categorizing a fictional request, finding the approved answer, drafting a response, and escalating an unknown issue. Demonstrate a failed critical verification step that blocks sign-off even if the total rubric score is high.

Acceptance Check

Every module has practice and a pass criterion; critical steps can block sign-off; missing policies are clearly marked. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1177. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Role adaptation: Rebuild the modules for [role] using these approved tasks and sources: [paste].

Time limit: Fit the learning path into [hours/week], identifying prerequisites and deferred modules.

Accessible format: Adapt the manual to [accessibility needs] with alternative practice and assessment formats that test the same skills.

Policies are invented. Replace unsupported rules with pending-source labels and request the approved document.

Reading equals passing. Require a demonstrated task and recorded assessor evidence.

Training is overloaded. Prioritize prerequisites and essential independent-work tasks first.

Assessment is subjective. Rewrite criteria as observable behaviors with examples for each score.

P035 Catering consultations and venue appointments

Procedures, people and delivery

Source project: Scheduling Project. Master pp 1046, 1255-1256, 3248. Restaurant application below is an editorial adaptation.

Restaurant application

Schedule tastings, catering consultations, private-room visits or service appointments with resource and conflict checks.

Source based lesson

Organize service appointments, staff availability, confirmations, and cancellation records.

Gather and build

Service duration, staff availability, timezone, buffer, cancellation. Model availability before bookings; prevent overlaps; record confirmation and cancellation separately.

Check before relying on it

Test simultaneous bookings and a timezone change. Rehearse booking, rescheduling, cancellation, and confirmation with sample clients.

Source based practice with fictional inputs

A booking starts at 10:00, lasts 45 minutes, and needs 15 minutes of cleanup.

Worked result: The resource is occupied until 11:00. Include the full occupancy interval in conflict checks.

Try a changed condition: The next booking starts at 11:00 with no additional setup requirement.

Check your answer: It does not overlap under the stated interval rule.

Restaurant boundary

Original is service-appointment scheduling, not ready-made restaurant tables, seating turns or staff rostering. Those require additional specified and tested rules.

P035 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

Which services and resources are booked? What durations and buffers apply? What timezone and availability matter? What cancellation and confirmation terms are approved? Is a simple manual calendar sufficient?

Structure

Bookings: BookingID, ClientRef, Service, Resource, StartDateTime, EndDateTime, Status, Confirmation, Notes. Availability: Resource, Date, Start, End. Add Services and DailyView.

Build Requirements

Use Requested/Confirmed/Completed/Cancelled/No-show status. Start and end are full date-times in one explicit timezone; show a separate note for client timezone when needed. A requested slot is not confirmed until capacity is checked. Buffers count toward occupied time.

Provide an overlap test for the same resource: another active booking overlaps when its start is before this booking's end and its end is after this booking's start. Exclude the current BookingID and Cancelled records. Adjacent appointments touching at a boundary do not overlap unless buffers extend occupancy. Give exact COUNTIFS or SUMPRODUCT formulas and flag invalid end<=start.

Create a daily view with service, time, resource, status, and safe client reference. Draft confirmation, reminder, reschedule, and cancellation messages based on supplied policies. Do not send messages, charge fees, or claim calendar synchronization.

Test resource A booked 09:00-10:00: 09:30-10:30 conflicts; 10:00-11:00 does not without a buffer. A booking for resource B does not conflict. Add a manual reschedule process retaining prior timing in a change log and checking availability again.

Acceptance Check

Overlap, adjacency, cancellation, resource, and invalid-time tests produce the expected flags. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1257. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Add [buffer minutes] before and after each service.

Change the workflow: Adapt the booking rules to [staff/rooms/equipment] with explicit resource ownership.

Add a useful view: Create a weekly utilization view using available minutes and confirmed occupied minutes.

Check the inputs. Use full date-times, not text labels.

Check the workflow. Exclude cancelled bookings but retain their records.

Check the result. Check timezone and resource before judging a conflict.

P040 Restaurant-service time and project billing

Procedures, people and delivery

Source project: Time Management Project. Master pp 1051, 1275-1276, 3308. Restaurant application below is an editorial adaptation.

Restaurant application

Track actual work sessions and billable time for restaurant client work or catering projects, including overnight sessions and agreed rates.

Source based lesson

Capture work sessions, separate billable from nonbillable time, and prepare a reviewable billing summary.

Gather and build

Person, project, start, end, break, billable status, rate. Log actual sessions; prevent overlaps; obtain review before preparing a billing summary.

Check before relying on it

The same hour cannot be billed twice because it appears in two projects. Reconcile one weekly total to the original session rows.

Source based practice with fictional inputs

A session runs 09:00-12:00 with a 30-minute unpaid break.

Worked result: Worked time is 2.5 hours. Subtract the stated excluded break and use consistent units.

Try a changed condition: A second entry duplicates the first hour of this session.

Check your answer: Resolve the overlap before totaling or billing; do not count the hour twice.

Restaurant boundary

Not a payroll, overtime, tip distribution or employment-law system. Billable totals are not proof of invoice payment.

P040 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

Which projects and activities matter? What timezone and billing rates apply? What rounding and minimum rules are agreed? Who approves time? How should unbilled and invoiced work be distinguished?

Structure

TimeLog: EntryID, Date, ProjectID, Task, StartDateTime, EndDateTime, BreakMinutes, Hours, Billable, Rate, Amount, InvoiceRef. Add Projects and Summary.

Build Requirements

Hours H2: =IF(A2="","",IF(OR(E2="",F2="",G2=""),"",IF(OR(F2<E2,G2<0,G2>(F2-E2)*1440),"",(F2-E2)*24-G2/60))) Use full date-times for overnight work. Blank breaks mean unconfirmed, while zero is explicit no break. Flag invalid ranges separately rather than hiding errors. Billable is a true checkbox. Amount is Hours*Rate only for billable valid records; require an explicit rate, including zero if agreed.

Keep actual hours and any invoice-rounded hours separate. Implement rounding only from the supplied agreement and show its direction and increment. Do not round every session if the agreement rounds only the daily total. Give exact formulas and sample results.

Summarize actual hours by project, billable/nonbillable hours, reviewed unbilled amount, and invoiced entries. An InvoiceRef means linked to an invoice, not paid. Flag overlapping sessions for the same worker and duplicate EntryIDs.

Test 09:00-11:30 with 30 break minutes gives 2 hours; at a fictional 50/hour the amount is 100. A 23:00-01:00 next-day session with zero break is 2 hours. Add a weekly approval and correction log.

Acceptance Check

Both daytime and overnight examples return 2 hours; missing breaks stay unconfirmed; amount is 100. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1277. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt the tracker to [team members] with per-person overlap checks.

Change the workflow: Apply this agreed rounding rule: [paste], keeping actual time visible.

Add a useful view: Add an unbilled-work summary by client and period.

Check the inputs. Enter the next date for overnight end times.

Check the workflow. Keep invoice linkage separate from payment status.

Check the result. Check overlapping sessions and explicit break values.

P044 Restaurant operational risk register

Procedures, people and delivery

Source project: Risk Management Project. Master pp 1055, 1291-1292, 3356. Restaurant application below is an editorial adaptation.

Restaurant application

Document supplier interruption, data loss or other actual operational risks, owners and control evidence.

Source based lesson

Identify operational risks, assign owners, plan responses, and review whether controls reduce exposure.

Gather and build

Risk event, cause, consequence, owner, control, review date. Write risks as events; prioritize consistently; assign an action and evidence that the control works.

Check before relying on it

A risk score is a prioritization aid, not a measured probability unless justified. Run one control test and update the risk record.

Source based practice with fictional inputs

A risk entry says supplier problem, high, no owner.

Worked result: It lacks a defined event, consequence, warning, and response. Write cause-event-effect, owner, trigger, and response before ranking.

Try a changed condition: The supplier misses a confirmed delivery date.

Check your answer: Record the current issue and response while separately reviewing future risk.

Restaurant boundary

Not a food-safety/HACCP plan or insurance assessment. Planned controls cannot lower current exposure until verified; scores are ordinal prioritization aids.

P044 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

What objectives and processes are in scope? What could prevent delivery? Which incidents or evidence exist? Who owns each risk? What scoring definitions and review cadence will you use?

Structure

Risks: RiskID, Objective, Cause, Event, Impact, Likelihood, Severity, Score, ExistingControl, Response, Owner, DueDate, Status, ResidualLikelihood, ResidualSeverity. Add Actions and ReviewLog.

Build Requirements

Write each risk as cause, uncertain event, and consequence. Separate risks from issues already occurring. Define likelihood and severity on a 1-5 scale with concrete anchors chosen for the business. Score=Likelihood*Severity only when both are valid; use it for prioritization, not as a precise probability or monetary expected loss.

Record response type Avoid/Reduce/Transfer/Accept, action, accountable owner, deadline, evidence, and review trigger. Residual score uses reassessed likelihood and severity after controls are actually implemented; a planned action alone does not justify a lower score. Retain original scores and dated reviews.

Give exact formulas and validation, open-action and overdue views, and a summary by risk owner. Flag missing owners and unsupported residual reductions. A high score needs a defined escalation route, but thresholds are the user's governance choice, not a universal standard.

Test likelihood 4 and severity 3 gives score 12. After a verified control, reassessment 2 and 3 gives residual 6. If the control is only planned, keep current exposure 12. Include a workshop agenda and five fictional risk examples tied to small-business operations without claiming exhaustive coverage.

Acceptance Check

The sample scores 12 then 6 only after verification; all open responses have owners and dates. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Editorial implementation clarifications

Apply these explicit additions or scope limits to the source brief before building. They resolve missing fields or boundaries; the original master has not been edited.

D12: Append a computed ResidualScore after existing fields, guarded for valid reassessed inputs and implemented-control evidence. This is a minor explicit-schema repair, not a changed risk method.

Source extensions and troubleshooting

Master p1293. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt scoring anchors to [business impact categories].

Change the workflow: Add an escalation rule using our approved threshold [value].

Add a useful view: Create a review log showing original, current, and residual assessments without overwriting history.

Check the inputs. Do not treat ordinal scores as exact probabilities.

Check the workflow. Separate an active issue from a future risk.

Check the result. Require evidence before crediting a planned control.

P045 Restaurant service continuity plan

Procedures, people and delivery

Source project: Business Continuity Project. Master pp 1056, 1295-1296, 3368. Restaurant application below is an editorial adaptation.

Restaurant application

Prepare approved fallbacks for POS/system outage, supplier loss or staffing disruption and rehearse on paper.

Source based lesson

Prepare a practical response and recovery plan for interruptions to critical business activities.

Gather and build

Critical service, disruption scenario, recovery target, contacts, backups. Map essential dependencies; choose a manual fallback; rehearse a small outage on paper.

Check before relying on it

Having a backup does not prove it can be restored. Test recovery of one non-sensitive sample file and record the result.

Source based practice with fictional inputs

A backup copy opens, but its linked source table is absent.

Worked result: The complete recovery test has failed. Include required dependencies and verify a meaningful result in a safe test copy.

Try a changed condition: A newer backup includes the table but has not been tested.

Check your answer: Its completeness is still unverified until the relevant restore checks pass.

Restaurant boundary

Do not invent hazardous recovery actions or food-safety thresholds. Keep safety/official instructions first; distinguish recovery goals from demonstrated capability.

P045 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

Which activities must continue? What interruption scenarios matter? What dependencies and backups exist? Who can activate the plan? What recovery objectives are realistic and supported?

Structure

Critical-activity assessment; activation criteria; role/contact tree; scenario playbooks; recovery checklist; communication drafts; exercise plan; revision log.

Build Requirements

Identify critical activities, maximum tolerable disruption as a user decision, required people/tools/data/suppliers, and minimum viable service. Distinguish recovery-time objective from proven recovery capability. Record assumptions and tests still needed rather than promising uninterrupted operations.

Write activation triggers, decision authority, alternate owner, initial assessment, and escalation. Provide scenario playbooks for loss of premises, key staff unavailability, supplier interruption, and system outage, adapted to actual dependencies. Prioritize personal safety and official emergency directions; do not invent technical recovery commands or hazardous response procedures.

Create a recovery checklist with owner, prerequisite, evidence, and return-to-normal approval. Backup records need location/reference, scope, owner, last successful test, and restoration instructions maintained by the responsible technical person. A backup existing is not evidence it can restore successfully.

Draft internal status, customer service-impact, and supplier coordination messages with confirmed facts, next update time, and contact. Avoid unsupported recovery promises. Store sensitive contact details in a restricted appendix; use fictional examples in AI chats.

Run a tabletop exercise: the primary work system is unavailable for a business day. Identify the minimum service, alternate process, data reconciliation, and decision points. Record gaps, owners, deadlines, and a retest. Include a one-page activation card and review after exercises, incidents, and major dependency changes.

Acceptance Check

Activation authority, critical dependencies, recovery evidence, communication drafts, and an exercise plan are complete. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1297. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt playbooks to [critical service] and these actual dependencies: [list].

Change the workflow: Create a one-page activation card for the responsible team.

Add a useful view: Add an exercise log with findings, actions, owners, and retest dates.

Check the inputs. Recovery objectives must be separated from demonstrated capability.

Check the workflow. Verify alternate access and restoration through a controlled test.

Check the result. Keep communication factual when the recovery time is unknown.

P046 Restaurant meetings and accountable actions

Procedures, people and delivery

Source project: Meetings Project. Master pp 1057, 1299-1300, 3380. Restaurant application below is an editorial adaptation.

Restaurant application

Record pre-service or management decisions and follow-up, for example the source's menu-update action, with owners, due dates and closure evidence.

Source based lesson

Make meetings purposeful, capture decisions, and carry accountable actions into the next review.

Gather and build

Meeting purpose, decision needed, owner, due date, prior actions. Circulate a short agenda; record decisions separately from discussion; review incomplete actions next time.

Check before relying on it

Every action needs one accountable owner and a verifiable output. Cancel or shorten a meeting that has no decision or coordination need.

Source based practice with fictional inputs

A meeting discusses three proposals but approves only one; two tasks are assigned.

Worked result: Record one decision, two unresolved proposals, and two owned actions. Separate discussion, decisions, and actions with evidence of agreement.

Try a changed condition: Attendees dispute the wording of the one approval.

Check your answer: Mark it unresolved and obtain clarification before acting on the disputed scope.

Restaurant boundary

A proposed action is not an accepted commitment. Keep confidential employee or guest details out of shared summaries.

P046 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Intake

What outcome requires a meeting? Who decides and who contributes? What preparation is needed? How long is available? Where will decisions and actions be reviewed?

Structure

Agenda template; pre-read request; decision log; action register; minutes template; follow-up message; recurring review routine.

Build Requirements

Create an agenda with outcome, item owner, time allocation, pre-read, decision required, and desired output. Total agenda minutes must fit the meeting duration, including opening and close. If an item needs only an update, offer an asynchronous note instead of using decision time.

Decision log fields: DecisionID, Date, Question, Options, Decision, Rationale, DecisionMaker, Evidence, RevisitTrigger. Action register: ActionID, DecisionID, Action, Owner, DueDate, Status, CompletionEvidence. One action has one accountable owner; collaborators may be listed separately. Status Not started/In progress/Blocked/Done/Cancelled.

Supply a complete fictional 30-minute project check-in: 3-minute opening, 7-minute progress exceptions, 12-minute decision, 5-minute actions, 3-minute close. Draft minutes that distinguish discussion, decision, and action. Avoid attributing agreement to someone unless recorded.

Create an action-review routine: confirm evidence for Done, name blockers, renegotiate dates visibly, and carry open actions forward without new duplicate IDs. Provide a concise follow-up message with decisions, owner/date actions, and correction deadline. Include a meeting-effectiveness reflection: was the intended decision made, and what preparation was missing?

Acceptance Check

The sample agenda totals 30 minutes and notes separate decisions from actions and discussion. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1301. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt the agenda to [meeting purpose] within [minutes].

Change the workflow: Create a decision-only version using these pre-read materials: [list].

Add a useful view: Add an overdue-action review table that preserves original IDs and date changes.

Check the inputs. Agenda time must include transitions and the closing action review.

Check the workflow. Do not confuse discussion with an approved decision.

Check the result. Keep one accountable owner and verifiable completion evidence per action.

P051 Restaurant manager weekly workload and capacity planner

Procedures, people and delivery

Source project: Weekly Planning Project. Master pp 1062, 1319-1320, 3440. Restaurant application below is an editorial adaptation.

Restaurant application

Replace personal areas with service, purchasing, staffing coordination, marketing, and administration; retain tasks, fixed blocks, estimates, and actuals.

Source based lesson (restaurant wording adapted)

Build a reusable restaurant manager or project-week plan with task capacity, reserved time, actual time, and completion kept distinct.

Gather and build

Confirmed service, delivery, and meeting commitments; genuinely available planning time; priority tasks; duration estimates. Place immovable events first; add realistic task blocks; leave space for interruptions.

Check before relying on it

Count preparation and travel where needed. Compare planned and actual time at the chosen weekly review and revise the next week's estimates.

Source based practice with fictional inputs

A restaurant project week has eight genuinely available work hours; tasks require three, four, and two hours.

Worked result: Nine hours are planned, exceeding capacity by one hour. Reduce, defer, or split work using the actual available time.

Try a changed condition: A fixed commitment removes another two available hours.

Check your answer: Capacity falls to six hours; at least three planned hours must move or be reduced.

Restaurant boundary

Not a labor-law-compliant rota, payroll tool, or automatic staff assignment. Use actual availability; handle overnight shifts explicitly.

P051 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask for restaurant role or project; week-start date; locale and timezone; fixed service, delivery, and meeting commitments; actual available planning minutes; priority work areas; and whether a calendar or short next-action list is preferred.

Base Tabs And Columns

Tasks: A Task ID, B Date, C Task, D Area, E Priority, F Planned Minutes, G Actual Minutes, H Status, I Notes, J Due Flag. TimeBlocks: A Block ID, B Date, C Start Time, D End Time, E Activity, F Area, G Duration Minutes, H Conflict Flag. Lists: areas, priorities, and task statuses. Summary: Week Start in A2 and a real date in B2; selected-week task totals and a seven-day breakdown.

Task statuses: Not Started, In Progress, Done, Cancelled. Priorities: High, Medium, Low. A task has one target date; a time block represents time reserved on a specific day. IDs are permanent.

BUILD RULES Use one task per row. Put large projects into separate next-action tasks that have realistic durations. Actual Minutes is entered after doing the work; leave it blank when unknown. Zero is a deliberate zero. Do not automatically roll unfinished work into another week or mark tasks complete from time spent.

TimeBlocks uses real times with Date separate. The base version accepts only blocks that start and end on the same calendar date, with End Time later than Start Time. Flag overnight blocks as unsupported; do not enter midnight as a same-day end. For overnight restaurant work, first extend the model to full StartDateTime and EndDateTime fields and derive duration and conflicts from those full timestamps. Duration Minutes is (End Time minus Start Time) times 1440, with blank-input guards. Flag invalid durations. Flag overlapping blocks on the same date when another start is before this end and another end is after this start. Adjacent blocks sharing one endpoint are allowed. Explain that task durations and time-block durations may describe the same work and are not additive.

Tasks J2, filled through J1000: =IF(A2="","",IF(OR(H2="Done",H2="Cancelled"),H2,IF(B2="","No date",IF(B2<TODAY(),"Overdue",IF(B2=TODAY(),"Today","Upcoming")))))

WEEKLY VIEW Use dates greater than or equal to Summary B2 and less than B2+7. Provide a seven-day task view, planned task minutes, known actual minutes, tasks completed, tasks still open, and cancelled tasks. Define completion as Done divided by all non-cancelled tasks dated in the week; show N/A if there are none. Show missing actual durations separately rather than treating unknown as zero work.

Show fixed time blocks and free planning capacity separately. Ask me to enter an available-minutes value for each day. Compare planned task time with that capacity; highlight overloaded days. Do not infer that every unblocked hour is available for work. Provide a daily planned-versus-known-actual bar chart, a top-three-priorities area, and an end-of-week review with what worked, what moved, and one adjustment.

Tests

Tasks of 30, 45, and 60 minutes total 135; two Done among three eligible tasks gives 66.7%. Cancelling the third changes the denominator to two. A task dated B2+7 is excluded. Blocks 09:00-10:00 and 10:00-11:00 do not overlap; 09:30-10:30 conflicts with both. An empty week shows zero counts and N/A completion. End with setup and weekly-use checklists.

Editorial implementation clarifications

Apply these explicit additions or scope limits to the source brief before building. They resolve missing fields or boundaries; the original master has not been edited.

E03: For restaurant overnight use, prefer explicit StartDateTime/EndDateTime or clearly documented next-day-midnight representation; calculate actual date-time duration and overlap. Do not silently apply time-only subtraction to overnight shifts.

Source extensions and troubleshooting

Master p1321. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Use a simpler week: Reduce my weekly planner to a short daily task list and three weekly priorities. Preserve the seven-day date filter and completion definitions.

Change my week start: Make [day] my week-start day. Show how to update B2 and all labels without changing the seven-day boundary logic.

Rebalance an overloaded week: Review these planned tasks and available minutes: [paste]. Suggest what to move or reduce; do not change commitments without identifying them.

A task disappears. Check its date and the selected week. B2+7 belongs to the next week.

Time totals are doubled. Do not add task estimates to calendar blocks that reserve time for the same work.

Midnight or overnight blocks are outside this base time-only model. Use only same-date blocks with end later than start until a full-date-time extension is built and tested; never discard minutes around midnight to make a duration positive.

Everything shows overdue. Check real date values and status spelling. Blank dates should be No date; Done and Cancelled are closed.

P052 Opening, closing, and manager-admin routine occurrence log

Procedures, people and delivery

Source project: Daily Planning Project. Master pp 1063, 1323-1324, 3452. Restaurant application below is an editorial adaptation.

Restaurant application

Routine becomes an owner-approved non-hazardous restaurant checklist; copy approved step IDs and planned-time snapshots into each day/shift.

Source based lesson (restaurant wording adapted)

Design repeatable approved opening, closing, service-handoff, or manager-admin routines, then record planned and completed occurrences.

Gather and build

Approved steps, responsible role, operational trigger, shift/date boundaries, available minutes, mandatory checks. Attach the routine to an existing operational trigger. Define a smaller version only for optional steps; preserve every mandatory requirement.

Check before relying on it

A missed routine should not erase previous records or force an unrealistic catch-up. Try the approved routine for three occurrences and review one optional step that adds no useful result. Never remove a mandatory check to make the timing fit.

Source based practice with fictional inputs

An approved restaurant routine has a 45-minute available window; the proposed steps total 60 minutes.

Worked result: The routine exceeds the available window by 15 minutes. Use realistic durations and escalate a capacity gap. Reduce or reschedule optional work only; do not drop mandatory checks.

Try a changed condition: A ten-minute transition is also required within the same window.

Check your answer: Only 35 minutes remain for activities. The 60-minute proposal now exceeds the usable window by 25 minutes; resolve the gap without omitting mandatory steps.

Restaurant boundary

Do not invent commercial cleaning, safety, or food-handling steps. Repeated same-day shifts need unique occurrence identifiers rather than violating one StepID per date.

P052 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask which approved opening, closing, service-handoff, or manager-admin routine is in scope; which role owns it; its trigger and available minutes; the exact approved steps; shift/date boundaries; and what minimum non-safety-critical version is acceptable. Mandatory checks must not be removed to save time.

Base Tabs And Columns

Templates: A Step ID, B Routine, C Step Order, D Step, E Planned Minutes, F Anchor or Trigger, G Notes. DailyLog: A Entry ID, B Date, C Routine, D Step ID, E Step, F Planned Minutes Snapshot, G Actual Minutes, H Outcome, I Notes, J Minute Difference. Lists: routine names and outcomes. Summary: Selected Date in A2 with a real date in B2; daily measures and a seven-day review.

Outcomes are Not Started, Done, Partial, Skipped. A planned step copied into DailyLog begins Not Started. Use one occurrence of a Step ID per date in this base design. Give repeated daily occurrences separate Step IDs.

TEMPLATE AND DAILY LOG Help me arrange steps in a sensible order with approximate durations, transitions, and a clear anchor such as "after the approved opening handoff". Keep the full routine within my available minutes and offer a shorter minimum version. Use only approved routine steps; do not remove mandatory safety or verification checks to shorten the routine.

Templates is the reusable plan. Each day, copy the selected steps into DailyLog as VALUES, including the planned-minute snapshot. Historical steps and planned times must not change when I later edit Templates. Assign new Entry IDs, enter the date, and keep Step IDs stable. Do not use a live lookup for the historical planned-minute snapshot.

Validate nonnegative minute values. Actual Minutes stays blank until known. An intentional skipped step can have zero actual minutes, Outcome Skipped, and an optional reason. Flag duplicate Date plus Step ID combinations and invalid Step IDs. Do not treat a missing daily row as a completed or skipped step.

DailyLog J2, filled through J1000: =IF(A2="","",IF(OR(F2="",G2=""),"",G2-F2)) Positive values mean longer than planned; negative values mean shorter. Leave the variance blank when actual duration is unknown.

SUMMARY AND REVIEW For Selected Date, show scheduled steps, Done, Partial, Skipped, Not Started, total planned minutes, known actual minutes, and rows missing actual minutes. Completion is Done divided by all steps scheduled for that day, including skipped steps. Show N/A with no scheduled steps. A minimum-version day has its own chosen scheduled rows, not hidden failures.

Provide a routine-by-routine summary and seven-day completion chart. Label any partial-day data as incomplete. Compare total planned versus actual only for rows with actual minutes recorded, or visibly show the missing-duration count. Include a reflection with one helpful step, one friction point, and one change to test tomorrow.

Tests

Three steps planned for 5, 10, and 15 total 30. Actuals of 7, 10, and 18 total 35 and paired variance totals +5. With two Done and one Skipped, completion is 2/3. A missing actual duration produces a blank row variance. Changing Templates E must not change previous DailyLog F values. End with instructions for preparing a new day.

Source extensions and troubleshooting

Master p1325. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Create a minimum routine: Create a shorter routine only for approved optional steps. Preserve every mandatory step and its required verification. If the requested duration cannot fit the required work, flag the capacity problem and ask the authorized owner to resolve it rather than deleting required steps.

Fit a changing schedule: Adapt my routines for [shift pattern]. Use anchors and available time rather than assuming the same clock times every day.

Review the first week: Use these seven days of results: [paste]. Identify one repeated source of delay and suggest one small change for next week.

Old estimates changed. Historical planned minutes should be pasted values. Replace live template lookups with snapshots after checking the original plan.

Completion looks too high. Use all scheduled steps as the denominator, including Skipped and Not Started.

Blank actuals count as zero. Keep them blank and report missing durations. Only enter zero when you know no time was spent.

The same step appears twice. Use a distinct Step ID for each repeated daily occurrence; otherwise remove the accidental duplicate.

P053 Recurring service-practice and manager habit tracker

Procedures, people and delivery

Source project: Habits Project. Master pp 1064, 1327-1328, 3464. Restaurant application below is an editorial adaptation.

Restaurant application

Use observed tasks such as completing a handoff record or reviewing supplier exceptions; preserve planned, met, below target, and unknown states.

Source based lesson (restaurant wording adapted)

Track an approved recurring restaurant practice against scheduled opportunities while keeping completed, missed, unlogged, and unscheduled states distinct.

Gather and build

Approved practice definition, scheduled opportunities, observable target, log status. Define success observably; distinguish missed from not scheduled; review consistency over planned opportunities.

Check before relying on it

Unlogged and missed are different statuses. Investigate repeated misses against actual capacity and authorized requirements. Revise an unrealistic optional target rather than silently lowering a mandatory standard.

Source based practice with fictional inputs

An approved recurring restaurant practice is scheduled Monday, Wednesday, and Friday; completion is recorded Monday and Friday.

Worked result: Completion is 2/3 = 66.67% of scheduled opportunities. Match the denominator to the stated schedule.

Try a changed condition: Wednesday's entry is unknown rather than confirmed missed.

Check your answer: Show two completed, one unknown; do not silently label the unknown successful or missed.

Restaurant boundary

A completion percentage does not certify compliance or justify employee discipline. Unknown is not evidence a safety task was done.

P053 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask which approved recurring service or management practice is being tracked; why it matters; minimum target and unit; scheduled opportunities; start date; and review period. Use observed completion only; the log is not a compliance certification or employee score.

Base Tabs And Columns

Habits: A Habit ID, B Habit Name, C Minimum Target, D Unit, E Schedule Note, F Start Date, G End Date, H Reason. Log: A Entry ID, B Date, C Habit ID, D Planned, E Actual, F Target Snapshot, G Result, H Notes. Lists: habit IDs and allowed units. Summary: Week Start in A2 and a real date in B2; overall and per-habit results for seven days.

Planned is a checkbox. Use one Habit ID per date. Targets must be greater than zero in this minimum-target version. Define each habit so that more of the chosen measure means progress toward its target.

SCHEDULING AND HISTORY Create a Log row for each planned habit date before the week starts. Copy the target from Habits as a VALUE into Target Snapshot, so later target changes do not rewrite history. Unplanned rows can have Planned unchecked and Result Not scheduled. Missing planned rows cannot be inferred automatically: give a weekly schedule audit that compares intended days with created rows.

For habits about reducing something, help me reframe a positive success measure, such as a yes/no goal met, or propose a separately labeled maximum-target extension. Do not apply the minimum-target formula to a lower-is-better goal.

Log G2, filled through G1000: =IF(A2="","",IF(D2<>TRUE,"Not scheduled",IF(OR(F2="",F2<=0),"Set target",IF(E2="","Unlogged",IF(E2>=F2,"Met","Below target"))))) Actual 0 is a real recorded result; blank means Unlogged. Do not auto-fill blanks with zeros. Require valid dates, matching Habit IDs, nonnegative actuals, and positive target snapshots. Highlight duplicate Habit ID plus Date pairs and planned rows with no target.

Summary

For dates from B2 through B2+6, count planned records, Met, Below target, Unlogged, and invalid targets. Completion equals Met divided by planned records, including Unlogged and invalid rows in the denominator; show N/A if none are planned. Also show logging coverage as planned rows with a numeric Actual divided by planned rows. Label both measures separately.

Provide per-habit counts and completion rates, with a bar chart. Do not add unrelated units such as minutes and glasses into one actual-total figure. A habit scheduled three days should not be penalized for four unscheduled days. Keep streaks out of the base version; if requested later, define whether they count calendar days or consecutive scheduled occurrences and how missing entries behave.

Tests

Three planned reading records with target 10 and actuals 12, 10, and 0 yield two Met, one Below target, and 66.7% completion. Changing 0 to blank yields one Unlogged and logging coverage 66.7%, while completion stays 66.7%. An unchecked Planned row does not affect completion. An empty week shows N/A. End with setup, daily logging, and weekly review checklists.

Source extensions and troubleshooting

Master p1329. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Add another habit: Add [habit] with target [amount and unit] on [days]. Define a positive success measure, prepare its planned rows, and update summaries.

Change future targets: Change [habit] to [new target] starting [date]. Preserve all prior target snapshots and results.

Add a defined streak: Add a streak for consecutive scheduled occurrences, not calendar days. Define what happens for Unlogged, Below target, and Not scheduled before providing formulas.

A blank is marked as failure. Blank Actual should be Unlogged; 0 is a recorded below-target result. Keep both visible.

Unscheduled days lower completion. Only Planned rows belong in the denominator. Confirm the weekly schedule rows are complete.

Targets changed retroactively. Store Target Snapshot as a value, not a live link to Habits.

Different habits have meaningless totals. Compare completion percentages. Sum actual amounts only within the same habit and unit.

P054 Restaurant improvement goal and milestone dashboard

Procedures, people and delivery

Source project: Goals Project. Master pp 1065, 1331-1332, 3476. Restaurant application below is an editorial adaptation.

Restaurant application

Use measured goals such as shorter response time or more completed training demonstrations with baseline, target, date, unit, and owner.

Source based lesson (restaurant wording adapted)

Turn restaurant improvement goals into measurable outcomes, milestones, and next actions, with progress that works for increasing or decreasing a number.

Gather and build

Baseline, target, deadline, milestone, measurement unit. Define the result; split it into observable milestones; choose a controllable next action.

Check before relying on it

For decreasing measures, progress must move in the correct direction; do not reuse the increasing formula blindly. Review whether effort is producing the intended result rather than only activity.

Source based practice with fictional inputs

A restaurant improvement measure rises from baseline 20 to target 50; the current value is 35.

Worked result: Progress from baseline is (35-20)/(50-20) = 50%. Use the stated start-to-target distance and identify direction.

Try a changed condition: A different restaurant measure is intended to decrease from baseline 100 to target 60; its current value is 80.

Check your answer: Progress is (100-80)/(100-60) = 50%; the direction reverses.

Restaurant boundary

Validate direction, evidence, and capacity. Do not average unrelated goals or present linear progress as forecast.

P054 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask for up to five restaurant improvement goals; operational reason; measured baseline, target, unit, and direction; deadline; current evidence; owner; and available resources. Do not invent a target or promise an outcome.

Base Tabs And Columns

Goals: A Goal ID, B Goal, C Metric and Unit, D Direction, E Baseline, F Target, G Start Date, H Deadline, I Current Value, J Last Updated, K Progress, L Status, M Next Action, N Notes. Milestones: A Milestone ID, B Goal ID, C Milestone, D Due Date, E Status, F Completed Date, G Notes. Lists: goal statuses, milestone statuses, and directions. Dashboard: current goal progress, deadlines, and overdue milestones. Clearly label this a current snapshot.

Directions are Increase and Decrease. Goal statuses: Not Started, Active, Paused, Completed, Cancelled. Milestone statuses: Not Started, In Progress, Done, Cancelled.

GOAL DEFINITION For each goal, write the outcome, evidence of completion, baseline, target, date, and one next action. A qualitative goal can use a count of clearly defined completed milestones as its measure. If milestones have unequal effort, say so; a simple count does not measure effort.

Baseline, target, and current value must use the same unit. Increase requires Target greater than Baseline; Decrease requires Target less than Baseline. Flag direction mismatches. Keep baseline fixed unless I explicitly redefine the goal and document why. A blank Current Value is unknown; zero is a valid observation. Last Updated is manual so it reflects when I actually measured progress.

Goals K2, filled through K1000: =IF(A2="","",IF(OR(E2="",F2="",I2=""),"",IF(F2=E2,"Set distinct target",MAX(0,MIN(1,(I2-E2)/(F2-E2)))))) Format numeric K values as Percent. This visual progress is capped at 0% and 100%; show raw Current Value beside it so setbacks or overachievement remain visible. Explain that a linear progress fraction is not a forecast of future completion.

MILESTONES Link milestones by Goal ID and validate against existing IDs. Add concrete deliverables with dates and owners only if I want team use. Keep Completed Date manual. Flag duplicate IDs, unknown Goal IDs, missing due dates, and Done milestones without completion dates. Do not mark a goal Completed merely because its deadline arrived.

Dashboard

Show goals by status, current progress for each active goal, goals due in the next 30 days, past-deadline active goals, and open milestones due or overdue. Keep Paused goals visible separately. Exclude Completed and Cancelled goals from the active overdue list. Avoid averaging percentages across unrelated goals as if all goals have equal importance; offer an explicitly equal-weight average only if requested.

Tests

Baseline 0, target 20, current 5 gives 25%. Baseline 30, target 10, current 20 gives 50%. Equal baseline and target shows Set distinct target. Blank Current Value leaves progress blank. Current 25 against target 20 displays capped 100% while raw 25 stays visible. A Completed goal below its target must be flagged for review. End with a weekly progress review and next-action checklist.

Editorial implementation clarifications

Apply these explicit additions or scope limits to the source brief before building. They resolve missing fields or boundaries; the original master has not been edited.

E08: Use50 of200 pages =25%; label the correction explicitly.

Source extensions and troubleshooting

Master p1333. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Make a vague goal measurable: Rewrite [goal] as a clear outcome with an observable completion test, metric, proposed baseline question, target question, and first action. Ask before choosing restaurant targets for me.

Break it into milestones: Create five concrete milestones for [goal], based on my available time and deadline. State dependencies and what counts as done.

Revise a goal carefully: Help me revise [goal] because [reason]. Preserve the original baseline and history in a note; distinguish a new target from a new goal.

Decreasing goals show negative progress. Use (Current-Baseline)/(Target-Baseline). Both differences are negative when a decreasing goal makes progress.

Progress is blank. Confirm baseline, target, and current value are numeric and present. Zero is allowed.

The goal is above 100%. The displayed progress is capped, while the raw current value preserves overachievement.

A deadline passed but status did not change. Status is a deliberate choice. Review the goal rather than automatically calling it completed or failed.

P055 Owner or manager operating review workspace

Procedures, people and delivery

Source project: Life Organization Project. Master pp 1066, 1335-1336, 3488. Restaurant application below is an editorial adaptation.

Restaurant application

Replace life areas with chosen restaurant responsibilities; retain task/habit/goal source logs and clearly dated current indicators.

Source based lesson (restaurant wording adapted)

Create a weekly restaurant operating review of tasks, goal progress, and recurring practices, each linked to its own source record.

Gather and build

Restaurant responsibility areas, defined measures, source records, review date. Choose a few indicators; link them to their source logs; write one decision for the week.

Check before relying on it

Every indicator has a clear unit and reporting period. Remove a metric that never changes a useful decision.

Source based practice with fictional inputs

A restaurant operating review includes learning pages read, cash allocated to an equipment reserve, and tasks completed.

Worked result: These are separate measures with different units. Present each measure with its own definition and period.

Try a changed condition: One measure has no current data.

Check your answer: Show it as unknown, not zero or a negative rating.

Restaurant boundary

No universal restaurant health or staff performance score. Current goal values are not historical week-end values.

P055 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask for restaurant responsibility areas; locale/timezone; review week; tasks, goals, and recurring practices; source records; and measures to omit. Do not assign a universal restaurant-health or staff-worth score.

Base Tabs And Columns

Tasks: A Task ID, B Area, C Task, D Due Date, E Status, F Planned Minutes, G Notes. Goals: A Goal ID, B Area, C Goal, D Baseline, E Target, F Current Value, G Deadline, H Status, I Last Updated, J Progress. HabitLog: A Entry ID, B Date, C Area, D Habit, E Planned, F Met, G Logged, H Notes. Lists: areas, task statuses, and goal statuses. Dashboard: Week Start in A2 with a real date in B2; weekly task and habit summaries, a current goal panel, and a short review area.

Tasks: Not Started, In Progress, Done, Cancelled. Goals: Active, Paused, Completed, Cancelled. HabitLog E:G are checkboxes.

DATA DEFINITIONS Keep this workbook self-contained. If I already use another planner, help me choose one source of truth rather than entering the same records twice. Imports or live connections are optional later work, not part of the base build.

Tasks has one dated next action per ID. Planned Minutes is an estimate, not a measure of wellbeing or productivity. Goals uses one consistent unit per goal with a fixed baseline, distinct target, current value, and manual update date. Qualitative goals can use a clearly defined count of milestones.

Goals J2, filled through J1000: =IF(A2="","",IF(OR(D2="",E2="",F2=""),"",IF(E2=D2,"Set target",MAX(0,MIN(1, (F2-D2)/(E2-D2)))))) Show the raw value with its unit beside percentage progress. The formula supports targets above or below baseline and caps the display at 0% and 100%.

For HabitLog, create planned rows ahead of time. Logged means I recorded an outcome; Met means the planned habit was completed. Met checked with Logged unchecked is invalid and must be flagged. Unchecked Met plus checked Logged means recorded not met. With both unchecked, the outcome is unknown. Exclude unplanned rows from habit completion. Flag duplicate Date plus Habit combinations and missing dates or areas.

Dashboard

Use dates from B2 inclusive to B2+7 exclusive for weekly tasks and habits. Show tasks due, Done, open, and overdue as of today; define completion as Done divided by non-cancelled tasks due that week. Show planned habit records, valid logged records, Met, and completion as valid Met divided by planned records. Show N/A when a denominator is zero.

Current goal progress must be labeled Current, with Last Updated dates. It cannot represent historical week-end progress without a separate history log. Show progress by goal rather than inventing an unsupported combined operating score. Include a task-completion-by-area chart and habit-completion-by-area table, with denominators visible. Add a review section for one win, one difficulty, and the next action in each chosen area.

Tests

Three Done among five non-cancelled tasks gives 60%. Four valid Met among six planned habits gives 66.7%. Unplanned records are excluded. Met without Logged is flagged and not counted as a valid success. An empty week shows N/A rates. Changing the selected week must not relabel current goal values as historical results. Finish with a weekly review checklist.

Source extensions and troubleshooting

Master p1337. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Choose my restaurant work areas: Use these areas: [list]. Map existing records to them and simplify the dashboard to the measures that help me choose next actions.

Connect an existing planner conceptually: I already track [items] in [tool]. Propose which source should own each record type and a manual update routine that avoids duplicate entry.

Simplify the dashboard: Limit the overview to tasks due this week, three current goals, and habit completion. Keep missing-data indicators and period labels.

My score seems meaningless. Avoid combining unrelated areas into one unsupported operating score. Review individual measures and your own priorities.

Past weeks show today's goals. Goal values are a current snapshot. Add a dated progress history only if historical views are needed.

Habit completion is inflated. Count only Planned and Logged and Met records as successes; keep all planned rows in the denominator.

Data is duplicated. Choose one source for each record type. Reusing an ID should update a record, not create a second entry.

P062 Private post-service learning reflection and optional team-appreciation notes

Procedures, people and delivery

Source project: Reflection Project. Master pp 1073, 1363-1364, 3572. Restaurant application below is an editorial adaptation.

Restaurant application

Use one observed shift moment, why it mattered, optional next action, and weekly review; keep uncertain dates and neutral/difficult observations honest.

Source based lesson (restaurant wording adapted)

Keep optional factual post-service learning and appreciation notes without requiring private disclosures or forced positivity.

Gather and build

Observation date, factual service or teamwork event, operational significance, optional next action. Record a specific observation and its practical significance. Allow neutral or difficult outcomes and genuine gaps without collecting clinical or private-life details.

Check before relying on it

Skipping a day must not create a fabricated entry. Review a week for one useful teamwork practice or operational lesson to repeat or improve.

Source based practice with fictional inputs

An optional factual service-learning record has four dated observations and three blank days.

Worked result: There are four entries; blanks mean no entry. Preserve real gaps and allow neutral or difficult operational observations; do not infer private feelings or health.

Try a changed condition: The writer remembers an event later but not its exact date.

Check your answer: Label the uncertainty instead of assigning a fabricated precise date.

Restaurant boundary

Do not fabricate guest/staff stories, compel positive sentiment, expose private details, or send appreciation messages automatically.

P062 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask which factual service, teamwork, or learning observations should be reflected on; whether the record is private or shared; how long entries should take; which topics or identifying details must be omitted; and whether a weekly debrief is useful. Do not require personal feelings, health details, or forced positive judgments.

Structure

Daily page; 30 varied prompts; weekly reflection; optional appreciation note; four fictional examples; reuse instructions.

Build Requirements

Create a daily page with date, one moment noticed, why it mattered, and optional person/action associated with it. Allow neutral or difficult service observations and a clear option to skip. Do not ask for health details or portray reflection as treatment.

Write 30 distinct prompts grouped around service moments, teamwork, effort, workplace conditions, learning, and verified guest-service observations. Keep prompts concrete and avoid requiring positive judgments about harmful experiences. Supply four brief fictional entries in different styles, including one day when the writer simply records a neutral observation.

Create a weekly page asking which moments stood out, what patterns appeared, and whether the format felt useful. Do not score a person's feelings or require consecutive-day streaks. Add an optional appreciation note template that can remain private; drafting it does not mean sending it.

Specify a clean printable layout with enough writing space and a digital version using reusable headings. Include a date-based file naming method and privacy reminder limited to storing the record with access chosen by its author and authorized workplace purpose. Example (restaurant adaptation): 'The next shift received the updated pickup list before service; the handoff prevented a duplicate check' is sufficient without a grand lesson.

End with a seven-day gentle start plan and choices to shorten, change, or stop the routine based on the user's experience.

Acceptance Check

Thirty distinct prompts, four examples, reusable daily/weekly pages, and a skip option are included. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1365. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Rewrite the prompts in [plain/playful/reflective] language.

Change the workflow: Create a one-minute version with a single optional question per day.

Add a useful view: Add a monthly page for memorable moments without ratings or streak pressure.

Check the inputs. Allow neutral observations and mixed feelings.

Check the workflow. Avoid repeating the same prompt in slightly different words.

Check the result. Leave adequate response space and omit unnecessary personal questions and make reflection optional.

P074 Shared restaurant-team, vehicle, and event-resource calendar

Procedures, people and delivery

Source project: Family Planning Project. Master pp 1085, 1411-1412, 3716. Restaurant application below is an editorial adaptation.

Restaurant application

Map family participants to staff/event roles and driver to delivery/transport owner; preserve attendance versus transport conflicts and individual recurrence history.

Source based lesson (restaurant wording adapted)

Coordinate restaurant staff, events, shared vehicles, and preparation commitments, exposing conflicts in both people and resources.

Gather and build

People, events, timezones, transport, preparation tasks. Record who attends and who transports; add preparation reminders; review conflicts together.

Check before relying on it

Two simultaneous events may conflict even if different people attend because one driver is shared. Review the next seven days with the responsible restaurant team and confirm unresolved transport or preparation owners.

Source based practice with fictional inputs

Two restaurant event commitments overlap; different staff attend, but both require the same driver.

Worked result: There is a transport-resource conflict. Include shared transport and preparation constraints.

Try a changed condition: One attendee arranges separate confirmed transport.

Check your answer: Resolve that transport conflict without deleting the event history.

Restaurant boundary

Not labor-law, working-hours, or staffing eligibility software. Use confirmed availability and travel buffers; do not expose sensitive appointment reasons.

P074 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask which restaurant staff, vehicles, rooms, or event resources participate; confirmed and tentative commitments; timezone; transport/preparation owners; travel buffers; recurrence rules; and information that must remain private.

Structure

Events: EventID, Title, StartDateTime, EndDateTime, Person, Location, TransportOwner, PrepTask, Status, Notes. Add RecurringTemplates and WeeklyView.

Build Requirements

Use Confirmed/Tentative/Cancelled status and full date-times. The calendar is a manual planning tool unless a separate integration is explicitly built. Assign a responsible person for transport and preparation where needed; attendance and driving are different roles.

Create conflict checks when the same person or transport resource is needed at overlapping times. Use start<other end and end>other start; back-to-back events need user-chosen travel buffers. Cancelled events do not occupy time. Provide exact formulas or a clearly labeled manual conflict review if the chosen layout cannot support them.

Recurring templates generate new EventIDs for individual occurrences; cancelling one occurrence does not erase the whole history. Include a weekly team and resource review: fixed commitments, travel time, supplies, meals if relevant, and changes needing communication. Keep sensitive appointment details in a restricted reference rather than the shared title.

Test one driver assigned to events 16:00-17:00 and 16:30-17:30: conflict. Different drivers may resolve transport but the same attendee still cannot be at both. Create a printable week view and a change-notification checklist without sending messages.

Acceptance Check

Overlapping driver and attendee commitments are flagged separately; tentative and cancelled events are clearly labeled. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1413. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt the calendar to [restaurant participants and resources] and their sharing needs.

Change the workflow: Add [travel buffer] between locations and show unresolved transport gaps.

Add a useful view: Create a simple printable week view with preparation tasks and owners.

Check the inputs. Distinguish attendee conflicts from transport conflicts.

Check the workflow. Use full dates and times for overnight or multi-day events.

Check the result. Preserve occurrence IDs when rescheduling recurring events.

P075 Restaurant duty rotation and task-load coordination board

Procedures, people and delivery

Source project: Household Tasks Project. Master pp 1086, 1415-1416, 3728. Restaurant application below is an editorial adaptation.

Restaurant application

Map chores to approved recurring restaurant duties, eligible people to trained/authorized roles, and swaps to agreed handoffs; retain effort estimates and occurrence history.

Source based lesson (restaurant wording adapted)

Agree approved restaurant duty ownership and rotation, respecting training, capacity, and access while preserving occurrence history.

Gather and build

Approved recurring duties, observable completion standards, trained roles, capacity, accountable owner, rotation rules. Agree what completion means; allocate only authorized work; record each occurrence and preserve missed, swapped, reassigned, and completed states.

Check before relying on it

Fairness requires discussion of task effort and availability, not merely equal task counts. Review a week of actual effort and constraints with the team, documenting any authorized temporary reassignment.

Source based practice with fictional inputs

Two trained restaurant team members each receive three approved duties, but one set takes 20 minutes and the other 90.

Worked result: Equal counts do not show equal effort or an agreed fair arrangement. Discuss actual effort, capacity, authorized skills, and completion standards with the responsible team.

Try a changed condition: One team member's actual availability falls for a week.

Check your answer: Agree a temporary adjustment and review it, rather than silently assigning missed work.

Restaurant boundary

Equal task count is not equal effort; do not create automatic penalties or use the organizer to make ungrounded employment decisions.

P075 Source based build specification

Copy the shared starter, the matching document or spreadsheet setup, your restaurant context, and this specification together. Project-specific requirements are retained and adapted for restaurant use; repeated setup is consolidated. Apply the labeled implementation clarifications as part of the brief. This is not finished software.

Adaptation: restaurant-only context or a non-restaurant example has been revised. See the source and editorial index.

Intake

Ask which approved restaurant duties recur; completion standards; trained/authorized roles; capacity and access limits; accountable owners; rotation rules; and the agreed method for missed, swapped, or reassigned work.

Structure

Duties: DutyID, Task, DefinitionOfDone, Frequency, EstimatedMinutes, EligiblePeople. Assignments: AssignmentID, DutyID, DueDate, Owner, Status, CompletedDate, SwapNote. Add Rotation and WeeklyReview.

Build Requirements

Define duties clearly enough that everyone knows what completion means. Assign responsibilities through agreement, not gender or age stereotypes. Account for capacity and access needs. Keep one accountable owner for each occurrence while allowing agreed help.

Use Planned/Done/Skipped/Swapped status. Each recurrence gets a new AssignmentID; do not overwrite last week's record. A swap changes ownership with a note and agreement. Do not create automatic employee penalties, rewards, or disciplinary decisions from this coordination record.

Summarize assigned estimated minutes and completed occurrences per person. Explain that equal counts are not necessarily equal effort. Example: two 10-minute duties total 20 minutes, while one 30-minute duty is a larger estimated load despite fewer tasks. Include no-data handling and exclude skipped work from completed counts.

Create a rotation table, a printable weekly chart, and a short review script: what worked, what was unclear, what load changed, and what should be reassigned. Completion records are coordination aids, not a measure of a person's worth.

Acceptance Check

The chart preserves recurring history, agreed ownership, and effort estimates; the sample compares 20 versus 30 minutes. Give expected results for the worked example and one missing-input case. Finish with a short first-use checklist and a repeat-use routine.

Source extensions and troubleshooting

Master p1417. These are optional follow-on changes, not included features. Keep the restaurant scope and approval boundaries above.

Adapt the scope: Adapt assignments to these capacity and access needs: [list].

Change the workflow: Create a rotation that avoids assigning the same task repeatedly unless preferred.

Add a useful view: Show estimated minutes alongside task counts for a more useful load review.

Check the inputs. Clarify what Done means before tracking it.

Check the workflow. Do not equate equal task counts with equal effort.

Check the result. Keep occurrence history when tasks recur or swap owners.