Lesson Review and Index: Product Management Lessons 1 and 2

This index covers the two reviewed lessons, the two capstone case studies, and the recommended learning order. It links each topic to the guide that teaches it in depth. These lessons did not cover RAG, Replit or Nginx in depth, so those topics have their own guides rather than being forced in here.

1. Lesson 1: AI product costing and an n8n onboarding workflow

- **Define the job** of the AI feature: tone, length, audience. Send useful variables with each request.
- **Unit cost:** calculate cost from input and output tokens, then multiply by expected volume. Compare conservative, realistic and aggressive scenarios before deciding to charge, give a free allowance, or absorb the cost. *Guide: Tokens, context and cost.*
- **Define input, output, owner and cost** before using an LLM in any feature.
- **Employee onboarding:** an HR form captures hire ID, candidate name, email, department, role, manager and start date. Reference data sits in an access registry and a scheduling registry.
- **Three agents:** a Coordinator routes and tracks work; an Access agent decides and records access; a Scheduling agent handles meetings. *Guide: n8n multi-agent onboarding.*
- **Webhooks:** a doorbell. The workflow waits for an event instead of checking repeatedly. Use the test URL while building, then the production URL once active. Include a unique `hire_id` as the primary key.
- **Credentials and tools:** never connect a personal account to a practice automation. Consider permissions before connecting any tool. Build and test one small step at a time.

2. Lesson 2: Make, MCP and case-study thinking

- **Scenarios** are repeatable automations. Variables and settings let one flow serve many inputs. Check credit use, notifications and failure handling before you enable one. *Guide: Make.com.*
- **MCP** (Model Context Protocol) is a standard way for AI applications to connect to outside systems. A **tool** is a capability an agent can call. *Guide: Skills, subagents and agents.*
- **Repeated assignment expectations:** understand the user and the real problem first; define the goal and a success metric; split features into v1, v2 and cut; define the core loop, AI

strategy and feedback; create journey maps, a PRD and three key wireframes; measure events on a dashboard.

3. Capstone: Fireflies, meeting commitments

Part	Content
Problem	Commitments made in meetings show up in summaries but are not reliably assigned, owned, tracked or closed. The class figure: only 28 percent of extracted commitments reached a resolved state within 14 days.
User	Meeting attendees who owe or receive a follow-up.
Goal	Make every commitment visible, owned and resolvable, without becoming a full project-management tool.
Resolution states	A commitment is resolved only when it is **done**, **reassigned**, or **deliberately dropped**. Each state is explicit.
Success metric	Share of commitments resolved within 14 days (the baseline is 28 percent).
v1	Detect commitments from a transcript; show owner and due date; three resolution states.
v2	Reminders; reassignment suggestions; owner summaries.
Cut	Full project boards, task dependencies, time tracking.
Core loop	Meeting → commitment extracted → owner confirms → reminder → resolved state recorded.
AI strategy	Extraction with a confidence score; the owner always confirms before a commitment is saved.
Feedback design	One-tap confirm, edit or reject on each extracted commitment, so the model learns what matters to each team.
Boundaries	No full task management; no changes to the meeting platform itself.
Events	commitment_detected, owner_confirmed, reminder_sent, resolved_done, resolved_reassigned, resolved_dropped
Wireframe 1	Meeting summary with a list of detected commitments, each with owner and due date, and confirm, edit or reject buttons.
Wireframe 2	My commitments: list grouped by due date, with the three resolution states as clear buttons.
Wireframe 3	Weekly summary: open, resolved and dropped counts, plus a link to any item that is overdue.

4. Capstone: ixigo, the confirmed-seat problem

Part	Content
Problem	A traveller knows the origin and destination but has no confirmed seat on the preferred train or class.
User segments	Last-minute travellers; price-sensitive planners; travellers who must arrive by a fixed time.
Goal	Help the user discover, compare and choose viable alternatives, and understand each option honestly.
Decision factors	Price, travel time, arrival time, station convenience, confirmation probability and cancellation flexibility.
Uncertainty messaging	Show likelihood as a range, never a guarantee. Say what is unknown and why.
v1	Compare alternative trains and classes for the same route and date; show the six decision factors; show uncertainty.
v2	Nearby stations and dates; bus and flight options where data is available.
Cut	Redesigning rail inventory, timetables, quota rules, fares, payments, refunds, or support staffing.
Success metric	Share of users who choose an option and complete the journey without leaving to search again.
Core loop	Search → see viable options → compare on the six factors → choose → track the decision.
AI strategy	Rank options and explain trade-offs in plain language; never invent availability.
Feedback design	After each choice, ask whether the option was right, and log the reason.
Events	search_started, option_viewed, option_chosen, uncertainty_shown, option_changed_later
Wireframe 1	Results list with six factor columns and a clear confidence range per option.
Wireframe 2	Option detail: trade-offs in plain language, and cancellation terms.
Wireframe 3	Confirmation: choice saved, with a reminder of what remains uncertain.

5. The nine-step case-study framework

1. Problem framing
2. User research

3. Goal
4. Success metric
5. Scope (v1, v2, cut)
6. User flow and journey map
7. PRD
8. Wireframes
9. Metrics and event taxonomy (plus an optional experiment plan)

6. Recommended learning order

1. Tokens, context and cost (LLM basics and cost).
2. RAG and retrieval.
3. Skills, subagents and agents.
4. Webhooks, JSON and test versus production.
5. One n8n workflow with fake data.
6. Credentials, permissions and test accounts.
7. Make scenarios and MCP.
8. App building with Replit or Lovable.
9. One capstone, using the nine-step framework.

7. Trusted next steps

- Build one safe n8n webhook demo with fake data and a practice account.
- Learn MCP from its introduction, then the specification when you build.
- For Make, learn scenarios first, then MCP with the smallest permissions.
- Review any connector's permissions before you connect it.

Official sources: check current documentation for n8n, Make, Lovable, Replit and MCP before relying on a feature. Prices change; check the live pricing page on the day you decide.