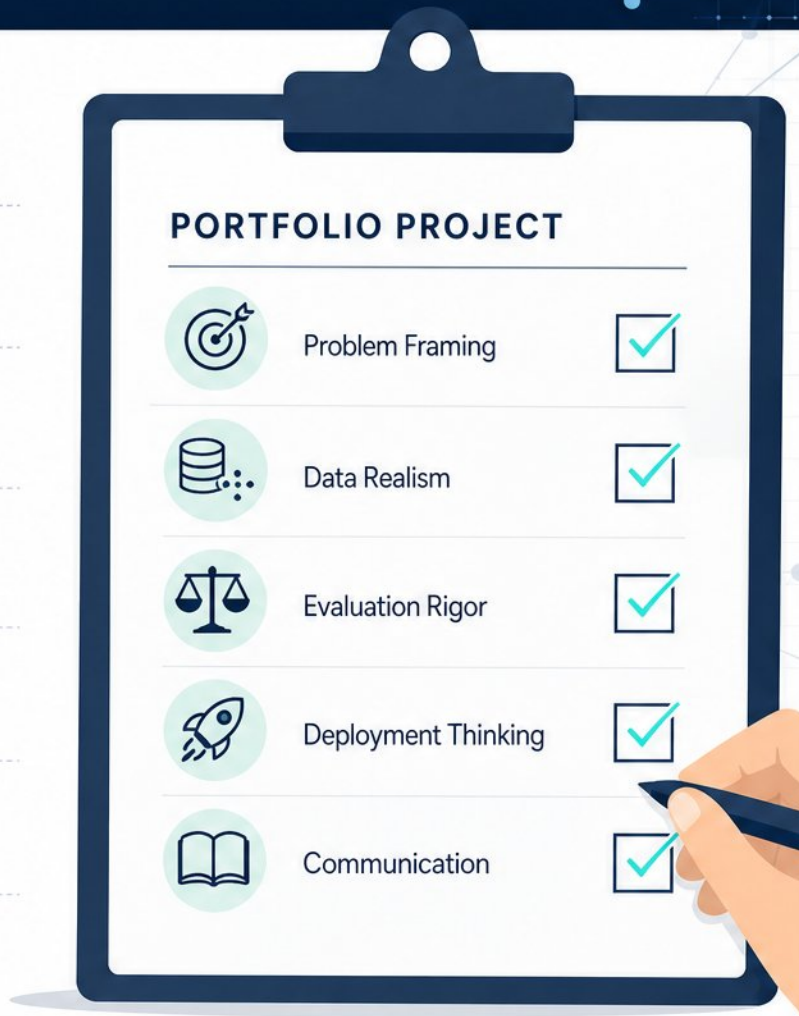
Building a Data Science Portfolio: Why It Matters and How to Set Your Goals

- A portfolio is verifiable proof of what you can build; a resume only claims.
- Reviewers give your GitHub roughly 90 seconds and ask one question.
- Can you turn a messy, ambiguous problem into something useful?
- Signal: two or three polished end-to-end projects with visible reasoning.
- Not signal: Kaggle ranks, certificates, and unmodified course notebooks.

What Reviewers Actually Look For in a Portfolio

- **Five dimensions:** problem framing, data realism, evaluation rigor, deployment thinking, communication
- **Problem framing:** a scoped question with a real stakeholder and decision
- **Data realism:** messy, self-sourced data beats pre-cleaned datasets everyone has seen
- **Evaluation rigor:** baselines, correct metrics for imbalance, proper validation, honest uncertainty
- **Deployment thinking:** what happens after model.fit() separates you from notebook-only applicants
- **Communication:** strong portfolios read like case studies a busy reviewer gets quickly
- **Dashboards support** but should not dominate — too many read you as an analyst



THE 90-SECOND REVIEWER'S EYE

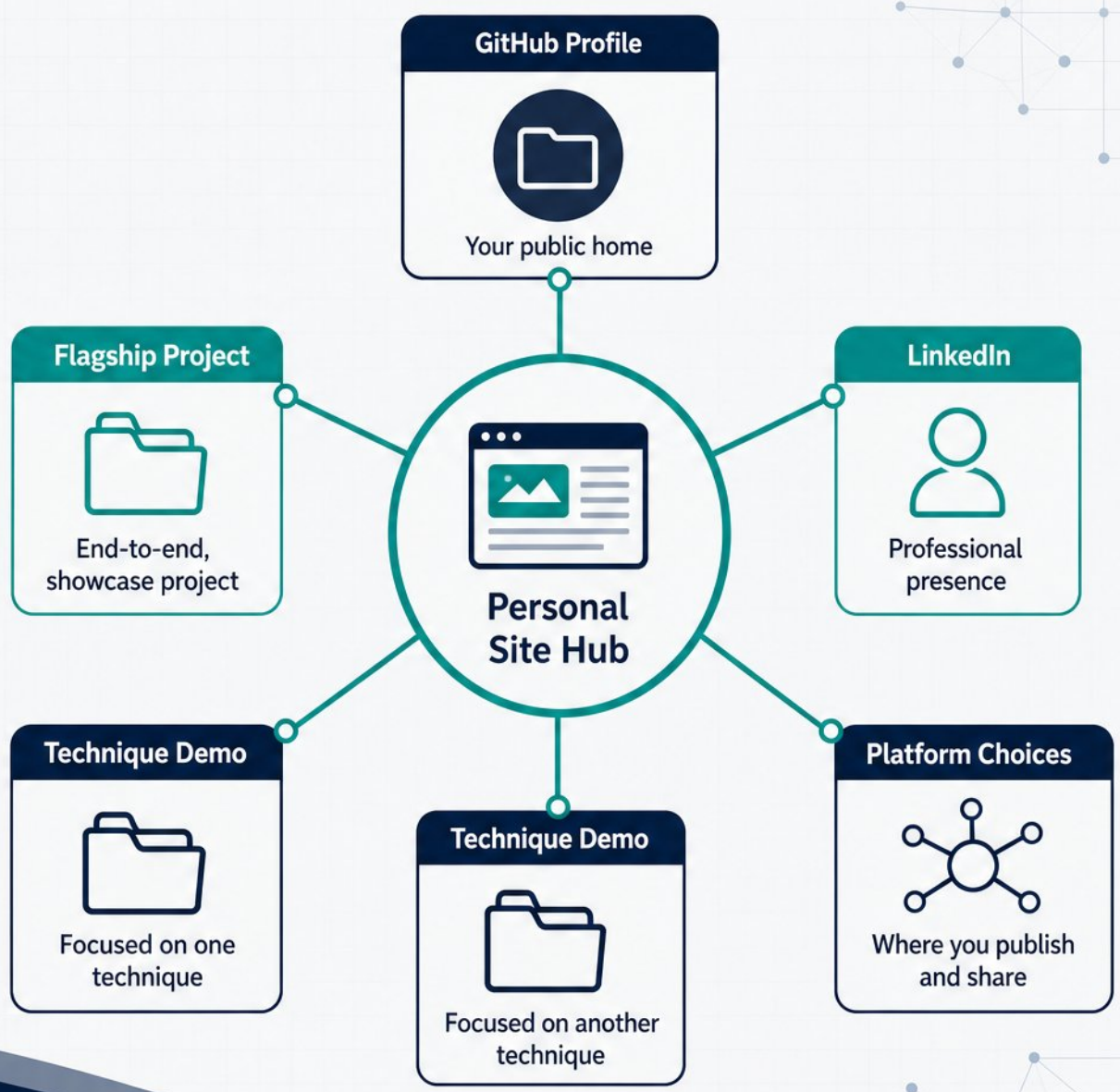
SCREENING
Does it look relevant at a glance?

SCAN
Do the five dimensions hold up?

SHORTLIST
Would I read this more closely?

Portfolio Architecture: Projects, Profiles, and Platform Choices

- Practical structure: GitHub profile, two to four polished projects, LinkedIn, personal site hub
- Recommended mix: one flagship end-to-end project plus one or two focused technique demos
- One clean repository per project, never a mega-repo with twelve folders
- First-impression surfaces: pin six role-relevant repos, profile README, remove stray forks
- Omit tutorial clones (Titanic, Iris, MNIST), unmodified coursework, employer data
- Platform choices: GitHub, Streamlit, Hugging Face Spaces, Gradio, Tableau Public
- Position toward the role you want; a focused profile beats a generalist one



Selecting Projects That Signal Employability

PROJECT SPECIMENS & WHAT THEY SIGNAL

Churn Prediction

Signals:
- Classification
- Feature Engineering
- Business Understanding
- Customer Insights

Forecasting

Signals:
- Time Series Thinking
- Trend & Seasonality
- Modeling
- Decision Support

NLP Sentiment

Signals:
- Text Processing
- Sentiment Analysis
- Feature Representation
- Unstructured Data

Deployed App

Signals:
- End-to-End Build
- Deployment Mindset
- Reliability Thinking
- User Impact

Deep EDA

Signals:
- Data Exploration
- Pattern Recognition
- Data Quality Thinking
- Problem Framing

- Start from skills and problems in real job postings you target
- Churn prediction, forecasting, NLP sentiment, and a deployed app
- Cover EDA, modeling, A/B testing, dashboards, and data engineering basics
- Source messy, ethically collected data via APIs or polite scraping
- Frame each project with a stakeholder and the decision it supports
- Scope to two-to-six weeks; finish smaller over abandoning ambitious
- Build where you have domain expertise; specialists stick in memory

From Raw Data to Reproducible Repository

- Standard layout: data/raw read-only, data/processed, notebooks for exploration, src for importable logic
- Data flows one way, raw to processed; every transformation is code, never manual edits
- Anything you run more than once belongs in a src/ function, not a notebook cell
- Pin dependencies, fix random seeds, use relative paths, add a data dictionary
- Atomic commits, proper .gitignore, and never commit data, credentials, or API keys
- Reproducibility test: clone, install, run one command, get the same results

Writing a README That Passes the 30-Second Scan

- Treat README as the product; code is supporting evidence
- Lead with title, one-line problem, and the punchline chart
- Story: data source, what was messy, key decisions, findings
- Copy-pasteable setup and run steps that actually work
- Limitations and next steps signal maturity louder than metrics
- Skip badge clutter; substance and readability come first

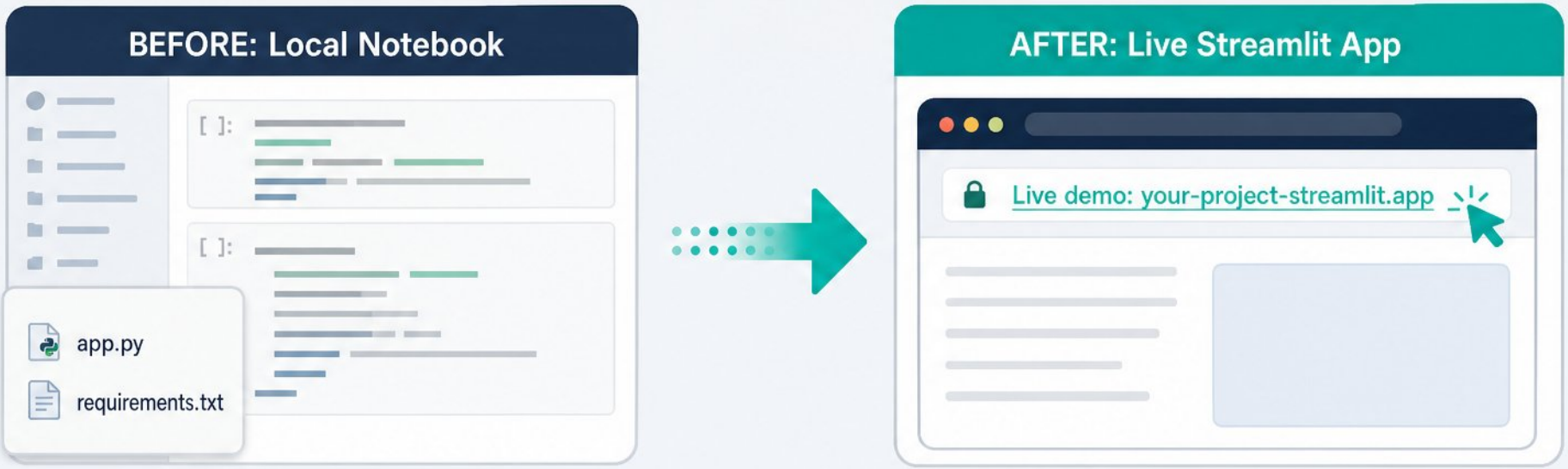
Documenting Decisions: Showing How You Think

- Reviewers can't separate intentional choices from accidents unless you write down why
- Explain big calls: dropping a column at 40% missing, choosing recall over precision
- Narrate the mess: show missing values, outliers, then how you handled each
- State what didn't work; failed models prove the final choice was deliberate
- Tie metric choices to the business problem, not a technical default
- Use markdown cells between code blocks so a stranger follows every decision
- A clean commit history says more than any accuracy number

Communicating Results: Visuals, Narratives, and Business Impact

- Lead with the result, not the tech stack
- Label axes, pick the right chart, cut chart junk
- 0.96 AUC means nothing without a baseline
- Three pitches: business, methodological, deep technical
- State limitations — bounded validity reads as senior judgment
- Prepare one-, three-, and six-minute versions of each story

Deploying a Live Demo (Without Overengineering)



Works on your machine

Runs for anyone, anywhere

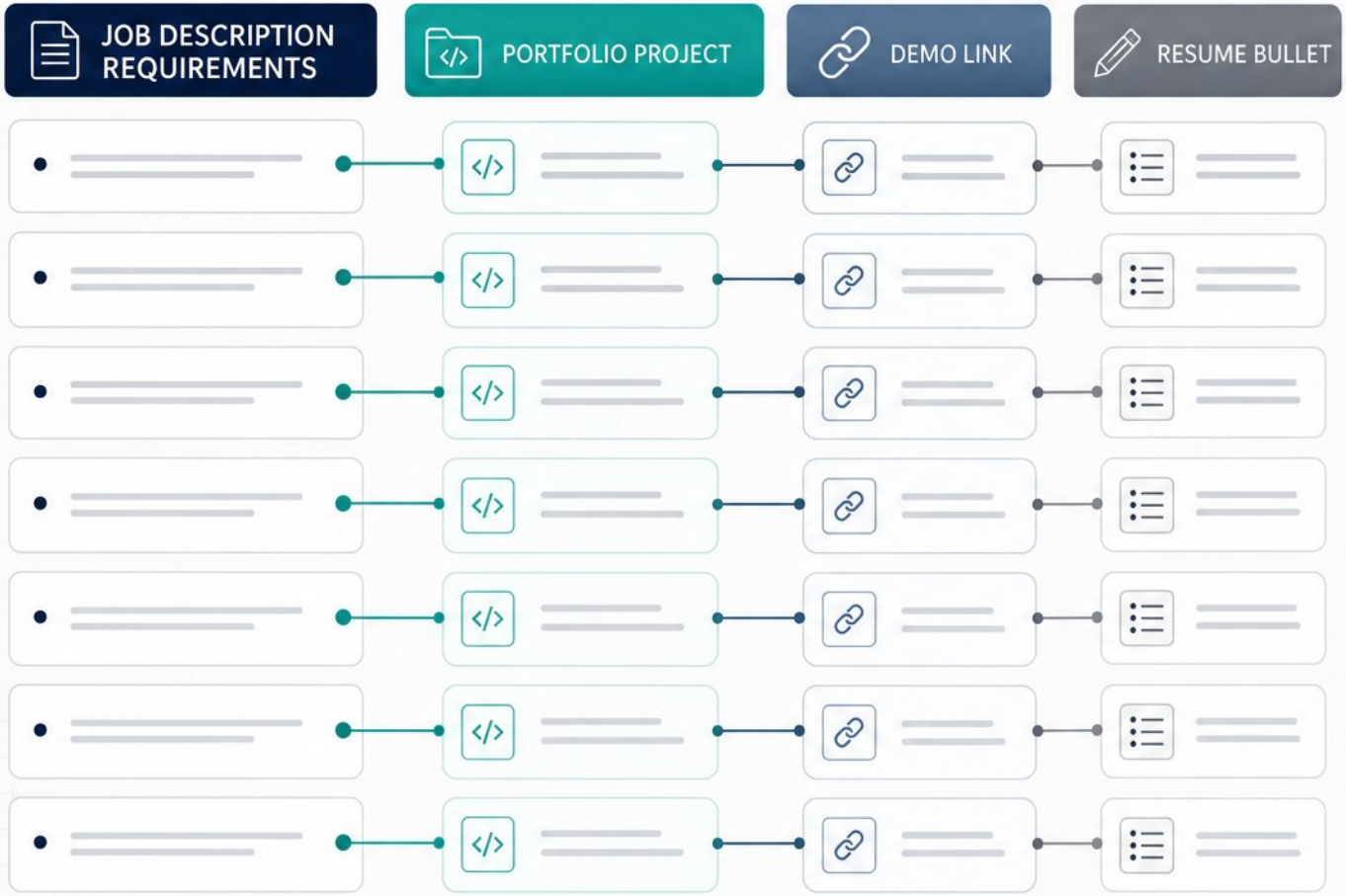
- Live deployment proves your work runs outside your laptop
- Free options: Hugging Face Spaces, Streamlit, Gradio, Vercel, Tableau Public
- Spaces basics: create a Space, add app.py and requirements.txt, push to redeploy
- Deploy one project, not five: pick your strongest business framing
- A well-documented notebook can beat an app no one can interpret
- Document setup in the README and never commit credentials
- Broken demo links can hurt more than having no demo at all

Building an Online Presence Without Overexposure

- Put exact role title and core tools in your headline, About, and experience
- Headline formula: role + two or three skills + one measurable outcome
- About section: problem you solve, decision it supports, stack, one proof point
- Feature your best work: pin projects, link live demos and repos
- Keep GitHub, LinkedIn, and your site telling one coherent story
- Never publish private data, proprietary code, or employer work without permission
- Community counts: meetups, Slack or Discord, open-source, write-ups

Your goal: Help the right people quickly understand what you do, the problems you solve, and the value you create—without oversharing.

Tailoring the Portfolio to Job Applications



TAILORING CHECKLIST

- Map each project to specific job description requirements.
- Resume bullet formula: decided X, used Y, changed Z.
- Link individual repositories, not just your GitHub profile.
- Mirror the posting's exact keywords in your headline and About.
- Click-test every link after PDF export before submitting.
- Cut tool dumping and vanity metrics; explain why you chose a tool.
- Get mentor feedback, then refine the story each application.

RAW INPUT
Job posting requirements and your project work

REFINE & ALIGN
Map, tailor, and test every link and message

POLISHED OUTPUT
Defensible portfolio that fits the role and gets noticed

Defending Your Projects in Interviews

- Interviewers already read your portfolio; they test how you think
- Four-layer answer: problem 30s, judgment calls 60-90s, results 30s, reflection 30s
- Prepare a data surprise, a modeling decision, and one failed attempt
- Always pair your metric with the baseline and the decision it informs
- Pitch three ways: business, methodological, and deep technical
- At your knowledge edge, admit it and explain how you would learn

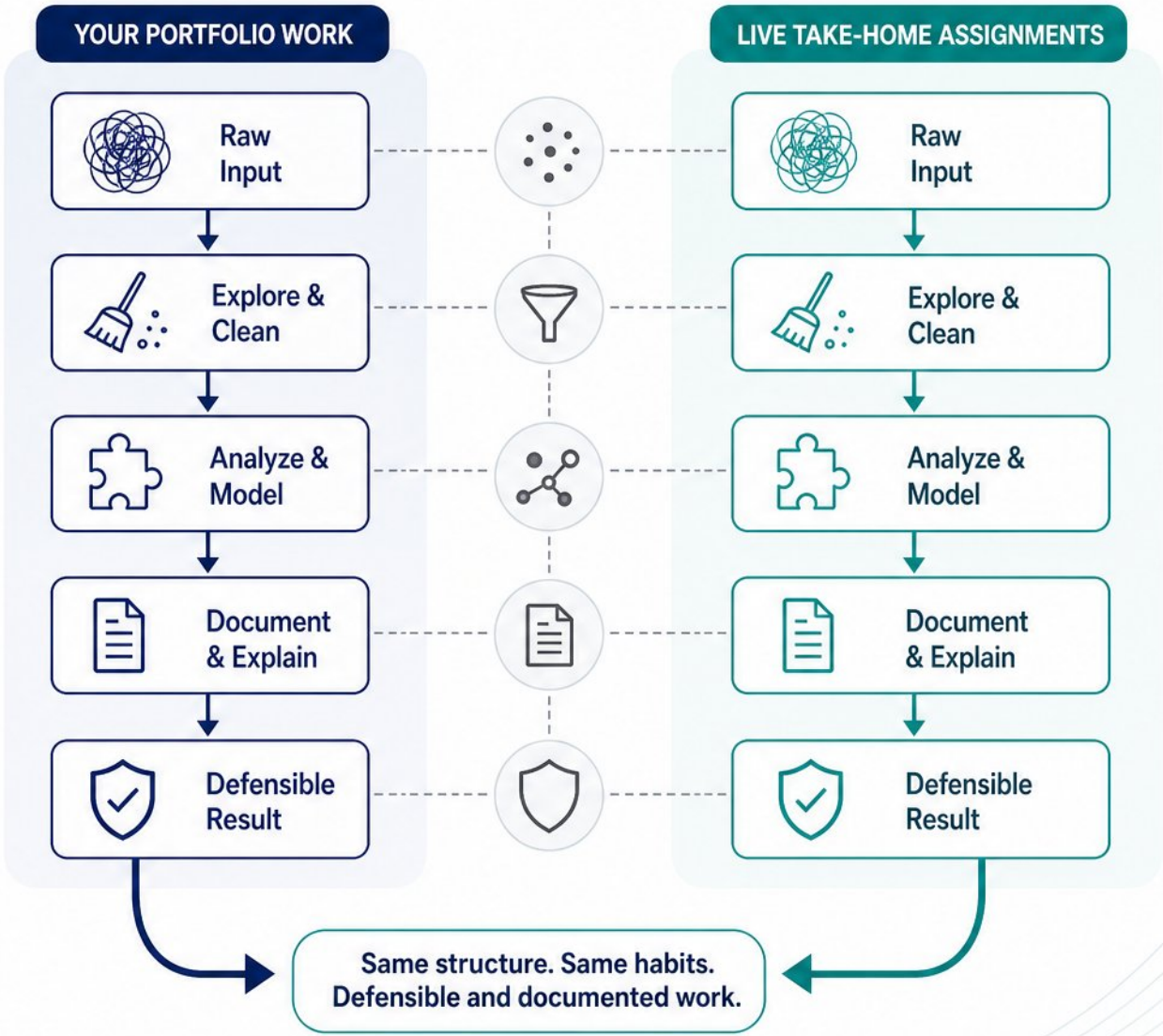
INTERVIEW FOLLOW-UPS:
What interviewers really want to know

Why did you choose this problem?	Your framing of the problem and business understanding
How did you approach it?	Your methodology, judgment calls, and trade-offs
What did you find and why does it matter?	The impact and the decision it informs
What would you do differently next time?	Your reflection, learning, and ability to improve

LAYERED PITCH STACK

Handling Take-Homes and Working Alongside Your Portfolio

- Junior interviews test fundamentals: SQL, Python, viz basics, project highlights — not deep learning
- Take-homes test the same skills; reuse your project structure and docs
- In case interviews, design the measurement before the model
- Expect hard follow-ups: why this dataset, why this model, how would you monitor it
- Treat each take-home as a small portfolio piece — document your reasoning and limits



Maintaining, Iterating, and Planning Your Next 90 Days

- Sustainable cadence: ship or refresh one project per quarter, not ten
- Deprecate honestly: fix dead links and broken demos before they undercut you
- Turn work projects into case studies only with permission
- Use avoided projects as a skill-gap tracker
- 90-day plan: ship one flagship, close one skill, polish one platform
- Keep GitHub, LinkedIn, and personal site telling one consistent story

REPOSITORY AS ARTIFACT

- /
  - README.md
  - projects
    - active
    - archive
  - case-studies
  - skills
    - target-gaps
    - notes
  - platform
    - site
    - github
    - linkedin
  - resources

README-as-Product

Clear purpose.
Transparent value.
Always up to date.

90-DAY ROADMAP

SHIP ONE FLAGSHIP

Build and ship one high-impact project.

Add to /projects/active

CLOSE ONE SKILL

Close one meaningful skill gap.

Document in /skills/target-gaps

POLISH ONE PLATFORM

Polish one platform surface.

Update in /platform

DEPRECATION LANE

Identify outdated or discontinued work. → Retire with intent: remove, archive, or redirect. → Keep the record clean and the story strong.

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/f73919d3/public