Master Resume Content Document — Che-Jung (Jerry) Chuang

0) Identity & Links

- **Name:** Che-Jung (Jerry) Chuang
- **Location:** College Park, MD (**Relocating Summer 2026**)
- **Phone:** 571-689-7318
- **Email:** jerry102102102@gmail.com
- **LinkedIn:** linkedin.com/in/che-jung-chuang-00886827a
- **GitHub:** github.com/jerry102102102
- **RL Brain Trainer Repo:** github.com/jerry102102102/RL_brain_trainer

Optional header variants (choose 1):

- *Robotics / Controls:* Robotics Engineer (Controls/Autonomy) | ROS2 | RL | Simulation-to-Deployment
- *SWE / Platform:* Software Engineer | Backend + Data Platform | ML Systems | Cloud-Native

---

1) Positioning Statements (Summary 模板，可依 JD 替換)

1A) Robotics / Controls / Automation（偏機器人）

Option A — industrial automation + controls + ML

Incoming M.Eng. in Robotics with strengths in control systems, ROS2 simulation, and reinforcement learning. Built sim-to-real oriented robotics training pipelines and safety-aware deployment patterns (fallbacks/watchdogs). Experienced in scalable system design and automation engineering; eager to apply hands-on robotics + software skills to industrial robotics and autonomy.

Option B — precision motion / mechatronics / RT Linux

Incoming M.Eng. in Robotics focused on precision motion control and mechatronic systems. Hands-on with ROS2/Gazebo and C/C++/Python control stacks; building an RL-driven joint control framework with safe fallbacks, timing/jitter profiling, and reproducible experiments. Comfortable with lab/PPE, safety interlocks, and traceable documentation.

Optional “micro-summary” (放在 Summary 下方一行，視空間使用)

- Controls + robotics software engineer: simulation pipelines, robust control design, and production-minded reliability (tests/guardrails/monitoring).

1B) SWE / Data Platform / MLOps（偏軟體）

Software Engineer with 1+ year of experience building cloud-native services and data platforms. Specialized in Python + Node.js (Nest.js), event-driven ETL on Azure Databricks, CI/CD + DevOps, and production ML integration for recommender systems and enterprise BI agents—delivering measurable impact (+20% CTR, +35% analytics success, -20% cloud costs).

---

2) Education（全量細節）

University of Maryland, College Park (UMD)

- **M.Eng. in Robotics** — Aug 2025 – May 2027 (projected)
- **Planned / target coursework (JD match pool):** Embedded Systems, ROS2, Motion Planning, Sensor Fusion, Reinforcement Learning, Control Systems
- **Coursework projects (completed / in-progress, as portfolio evidence):** ENPM662 (robot modeling / simulation), ENPM667 (control systems)

National Yang Ming Chiao Tung University (NYCU)

- **B.S. in Computer Science** — Sep 2019 – Jun 2023
- **GPA:** 3.7/4.3 (last 60 credits)

- **Relevant coursework:** Network Programming, Computer Networks, Probability and Statistics, Mechatronics Fundamentals

---

3) Work Experience（全量素材 + 可拆 bullets）

3A) STARK Tech — Taipei, Taiwan

Time: Mar 2024 – Jul 2025

Role variants (依 JD 替換 title)：

- Python Engineer
- Software Engineer (Python / Node.js)

Company context (coffee chat / 面試一行版)

Built systems across two major tracks:

1. **Content/news recommendation** powered by event-driven real-time pipelines + ranking/ML systems
2. **Enterprise BI / LLM agent platform** (text-to-SQL, visualization, HITL validation, automated regression)

3A-1) Real-time Streaming Pipeline / Data Platform（事件流 + ETL）

Resume bullets (short; pick 1–3)

- Designed and deployed a large-scale real-time automation pipeline (Azure EventHub → Databricks DLT → downstream services), processing **millions of events/day**; improved CTR by **20%**.
- Built streaming ETL with **SLO-driven alerting**, **idempotent updates**, and **regression tests** to ensure reliability at scale.

- Delivered ~**5 min end-to-end latency** and **100,000+ events/10 minutes** throughput; implemented **medallion (bronze/silver/gold)** patterns and **materialized views** for downstream apps.

Long-form deep dive (STAR + technical depth)

- **Situation:** Recommendation and analytics depended on near-real-time interaction signals (impressions, clicks, engagement). Volume was high; late / duplicated events could distort metrics and degrade model quality.
- **Task:** Build a pipeline that is **reliable at scale** (idempotent), **observable** (SLO/alerts), **recoverable** (replay/backfill), and **safe to iterate** (regression coverage).
- **Actions (architecture & reliability):**
  - Designed an event-driven ingestion flow using EventHub as the entry point and Databricks DLT for streaming transformations.
  - Implemented **idempotent update strategy** so retries/duplicates do not contaminate aggregates.
  - Added **SLO/SLI monitoring + alerting** (latency, backlog, failure rate) to detect degradation early.
  - Built **regression tests** to prevent silent logic/schema regressions during iteration.
  - Adopted **bronze/silver/gold** layering + aggregated/materialized views for stable downstream consumption.
- **Results:** Supported **millions/day** production event throughput while enabling measurable product impact (**CTR +20%**) and improving operational reliability.

Keywords bank (JD matching)

event-driven, streaming ETL, Azure Event Hubs, Databricks DLT, Delta, medallion architecture, idempotency, SLO/SLI, observability, regression testing, materialized views, backfill/replay, data contracts

---

3A-2) Enterprise BI LLM Platform（LLM agents + HITL + regression）

Resume bullets (short; pick 1–2)

- Led an enterprise BI agent platform integrating LLMs with **human-in-the-loop validation** and **automated regression tests**; improved query success by **35%** and reduced cloud costs by **20%**.
- Built an agentic BI assistant (text-to-SQL, visualization, reasoning) with **fail-fast validation** and nightly regression suites; improved analytics success **+35%** and reduced cloud cost **20%**.

Long-form deep dive (why this is "big tech SWE/Applied AI ready")

- **Problem:** BI agents fail in predictable ways (incomplete user intent, wrong metric/dimension/timeframe, SQL hallucination, mismatched visualization). Failures are costly: user trust drops and compute cost spikes due to retries.
- **Goal:** Make the agent **controllable, testable, and regression-safe** while improving success rate and lowering cost.
- **What you built (system behavior, not just "used LLM"):**
  - Implemented **HITL gating**: high-risk or incomplete queries are routed to human review or guided completion rather than executing unsafe outputs.
  - Built **fail-fast validation**: detect missing critical elements early, terminate or request clarification before expensive execution.
  - Established **nightly regression suite**: curated test set + automated runs to prevent quality drift during prompt/tooling iterations.
  - Reduced wasted compute by eliminating repeated invalid executions and increasing first-pass success.
- **Results:** Analytics query success **+35%**, cloud cost **20%**.

Keywords bank

LLM agents, text-to-SQL, guardrails, evaluation, regression testing, HITL, reliability engineering, cost optimization, prompt/tooling iteration, automated QA

3A-3) Recommender System (RL / DQN)（推薦系統 + 漂移/冷啟動）

Resume bullets (short; pick 1–3)

- Implemented reinforcement learning–based recommender (DQN) robust to **data drift** and **cold-start**; achieved **+20% CTR** and **+15% engagement** growth.
- Operationalized MLOps workflows (packaging/versioning, offline/online validation gates, staged rollouts) for recommender + BI assistant.
- Improved PR-AUC by **+0.07** in A/B evaluations; built related-news module using word2vec with keyword-weighted similarity.

Long-form deep dive (what makes this “production ML”, not “class project”)

- **Objective:** Improve ranking quality under real conditions: shifting content inventory (churn), distribution drift, and cold-start items.
- **Approach:** Used DQN-style policy learning for ranking/weight blending decisions, and added **fallback strategies** when signals are sparse or unstable.
- **Operationalization:** Versioning + offline/online gates + staged rollout mindset to reduce risk and keep learning changes reviewable.
- **Impact:** CTR and engagement improvements were measured and communicated as business outcomes.

Keywords bank

recommender systems, ranking, reinforcement learning, DQN, drift handling, cold start, A/B testing, PR-AUC, MLOps, offline/online evaluation gates, rollout strategy

3A-4) Backend Services / Microservices / DevOps（偏 SWE JD）

Resume bullets (short; pick 1–3)

- Owned and scaled backend microservices (Nest.js, Prisma, PostgreSQL on Azure); defined API/data contracts and schema evolution; tuned indexes for low-latency workloads.
- Established CI/CD: Dockerized services, added unit/integration tests and quality gates, implemented blue–green deployments with safe rollbacks.
- Built observability (structured logs, metrics, distributed tracing) with alerts and on-call runbooks to improve incident response and release reliability.

Long-form notes (signal: “production engineer”)

This block is a strong indicator of senior engineering habits:

- contracts & schema evolution
- performance (index/query tuning)
- deployment safety (blue-green + rollback)
- observability & operational readiness (alerts/runbooks/on-call)

Keywords bank

microservices, NestJS, Prisma, PostgreSQL, API contracts, schema migration, indexing, CI/CD, Docker, blue-green deployment, rollback, observability, tracing, runbooks

---

3B) Academia Sinica — Institute of Information Science (Taipei, Taiwan)

Role: Research Intern

Time: Jun 2022 – Oct 2022

Resume bullets (short; pick 1–2)

- Implemented graph-based resource allocation for Social IoT; improved robustness under dynamic topologies by **15%**.
- Performed scalability analysis and documented algorithmic trade-offs in internal technical reports.

Long-form deep dive

- **Problem:** Social IoT resource allocation is sensitive to topology changes; naive strategies degrade quickly when connectivity shifts.
- **Work:** Designed/implemented graph-based allocation logic and evaluated robustness under dynamic conditions; summarized trade-offs and scaling behaviors in research-style reports.
- **Result:** robustness improvement **+15%**.

Keywords bank

graph algorithms, resource allocation, dynamic topology, robustness evaluation, scalability analysis, research prototyping

---

4) Projects（全量細節 + 可替換 bullets）

> 這一章的目標：你未來投任何 JD，都能在這裡找到
>
> **(a) 一句話定位、(b) 可直接貼履歷 bullets、(c) 技術深挖/面試故事、(d) keywords、(e) repo/artefacts**

4A) RL Brain Trainer — ROS2 Control Framework for Sim-to-Real Joints

Time: Sep 2025 – Present

Repo: github.com/jerry102102102/RL_brain_trainer

One-liner (resume-ready)

Designed a modular sim-to-real pipeline that trains joint-level RL policies in Gazebo and deploys to ROS2 control nodes with safety-aware fallbacks/watchdogs.

Resume bullets (pick 2–4)

- Designed a modular training→deployment pipeline: learn joint policies in Gazebo, export deployable ROS2 controllers with safe-state fallbacks and watchdogs.
- Stabilized control-loop timing via executor/QoS profiling and callback tuning; added defensive handling for saturation/NaNs.
- Integrated system identification (step + sine sweep) to estimate actuator dynamics; benchmarked PID vs RL on rise time, overshoot, and tracking error.
- Built a CI-friendly test harness (multi-seed reproducibility) with unit tests; reduced manual tuning time by **40%**.

Deep technical notes (interview/coffee chat version)

- **Goal:** Not just “train an RL policy,” but make it deployable inside a ROS2 control-style stack with predictable behavior and safety mechanisms.
- **Architecture:**
  - Simulation: Gazebo + ROS2 robot model, sensors, dynamics
  - Training: PyTorch RL pipeline; designed to support repeatability (seeds, configs) and extensions (domain randomization where relevant)
  - Deployment: ROS2 control node/controller wrapper that can run policies safely and fall back to safe-state behavior
- **Reliability & safety patterns:**
  - Watchdog timeouts / heartbeat behavior
  - Guardrails for invalid outputs (NaN, saturation)
  - Timing/jitter profiling to avoid control instability caused by scheduling jitter rather than controller design
- **Evaluation discipline:**
  - Always keep a PID baseline

- Use measurable control metrics (rise time, overshoot, tracking error), not subjective “looks stable”

Keywords

ROS2, Gazebo, ROS2 Control, control loop timing, QoS, executor profiling, watchdog, safety fallback, system identification, PID baseline, sim-to-real, reproducible experiments, CI/testing

---

4B) UMD Robotics Coursework Projects (Controls & Modeling)

(這一段就是你要用來替換掉履歷上 “Gazebo 那個模糊專案” 的核心材料)

4B-1) ENPM662 — ROS2 + Gazebo Manipulator Motion Pipeline (IK → JointTrajectory → Controllers)

Artifact: ENPM662_Project 2_Final_Report.pdf

Context: Group project (robotics modeling / simulation engineering)

One-liner (resume-ready)

Built a waypoint-to-trajectory motion pipeline for a manipulator in ROS2 + Gazebo, including damped least-squares IK with numerical safeguards, trajectory unwrapping, and robust JointTrajectory publishing to ros2_control controllers.

What you built (high-level system)

A full pipeline from **desired end-effector waypoints** to **robot joint trajectories** that are safe to execute in simulation:

1. Generate/define waypoint targets (position + orientation)
2. Solve IK per waypoint using **damped least-squares (DLS)** with seed continuity
3. Post-process the joint sequence (unwrap angles, enforce prismatic bounds)
4. Publish a JointTrajectory message to the controller interface (with retry strategy)

5. Automate Gazebo bring-up (spawn robot, props, controllers, clock)

Technical deep dive (detail bank)

IK solver design (robust, production-minded):

- **Seed continuity:** reuse the previous waypoint's solution as the next initial guess to improve convergence and smoothness.
- **Prismatic constraint:** clamp the rack (prismatic joint) into **[-0.70, 0.70] m**.
- **Convergence criteria:** threshold **1e-4**, max **200 iterations**, default damping **1e-6**.
- **Numerical safeguards:**
  - Orientation error computed via rotation log-map / axis-angle style formulation
  - Clamp cosine into **[-1, 1]** before arccos to avoid NaNs
  - Small-angle fallback to avoid instability near zero rotation
  - Normalize joint axes when assembling Jacobians

Trajectory quality & execution stability:

- **Angle unwrapping (2π handling):** remove ±π discontinuities across waypoints so revolute joints move smoothly instead of "jumping" due to wrapping.
- **Controller publishing robustness:** publish `/arm_controller/joint_trajectory` with a configurable repeat strategy (`publish_repeat_count`, `publish_repeat_period`) to reduce transient drops and improve acceptance stability.
- **Fail-fast behavior:** structured logging and explicit failure handling if a waypoint does not converge, so debugging is deterministic.

Simulation bring-up automation (system engineering signal):

- Generated empty world, published URDF to `/robot_description` with `use_sim_time=true`
- Spawned robot and task props (e.g., tray/burner objects)
- Started controllers and bridged `/clock` for deterministic sim-time execution

Resume bullets (pick 2–4)

- Implemented DLS IK per waypoint with seed continuity, prismatic joint clamping, and convergence safeguards (max 200 iters, damping 1e-6, threshold 1e-4).
- Added numerical robustness to IK (rotation log-map error, clamped cosine, small-angle fallback, normalized Jacobian axes) to prevent solver instability.
- Smoothed joint sequences via 2π unwrapping; published reliable JointTrajectory commands to `/arm_controller/joint_trajectory` with retry strategy for controller acceptance.
- Automated Gazebo bring-up: robot + props spawning, controller startup, and `/clock` bridge for deterministic simulation runs.

Keywords

ROS2, Gazebo, ros2_control, JointTrajectory, IK (DLS), Jacobian, numerical stability, angle unwrapping, simulation automation, URDF/robot_description, deterministic sim-time, debugging/logging

---

4B-2) ENPM667 Final Project — Cart + Double Pendulum Crane (Nonlinear Modeling → Observer → LQG/LQI)

Repo: https://github.com/jerry102102102/ENPM667_Project2

Artifact: ENPM667_project_2 (1).pdf

Time: Dec 2025

One-liner (resume-ready)

Derived and simulated a nonlinear cart–double-pendulum crane model, linearized it for controller design, benchmarked Luenberger observers across measurement sets, and implemented LQG output feedback with integral augmentation (LQI) for tracking and constant disturbance rejection.

Deep technical notes (detail bank)

Nonlinear modeling (you did the “real” controls work):

- Formulated the system using **Euler–Lagrange** mechanics, obtaining coupled nonlinear equations of motion for cart position and two pendulum angles.
- Converted the dynamics into a first-order nonlinear state-space representation with state vector:

  x=[x,x˙,θ1,θ˙1,θ2,θ˙2]Tx = [x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]^Tx=[x,x˙,θ1,θ˙1,θ2,θ˙2]T

Linearization & feasibility checks:

- Linearized around an equilibrium configuration (cart at origin, pendulums hanging down, zero velocities, zero input).
- Performed controllability checks on the linearized system before committing to state feedback / observer designs.

LQR design with explicit weight choices:

- Designed LQR state feedback with

  Q=diag(1,0.1,10,0.5,10,0.5)Q = \mathrm{diag}(1, 0.1, 10, 0.5, 10, 0.5)Q=diag(1,0.1,10,0.5,10,0.5), R=0.01R = 0.01R=0.01
- Used stability and transient behavior to justify weight selection (not "random tuning").

Observer design & trade-off reasoning (this is a strong differentiator):

- Benchmarked **Luenberger observers** across multiple measurement outputs, e.g.:
  - y=xy = xy=x (cart position only)
  - y=(θ1,θ2)y = (\theta_1, \theta_2)y=(θ1,θ2)
  - y=(x,θ2)y = (x, \theta_2)y=(x,θ2)
  - y=(x,θ1,θ2)y = (x, \theta_1, \theta_2)y=(x,θ1,θ2)
- Designed observer gains via pole placement using speed factors

  α∈{4,8,12,20}\alpha \in \{4, 8, 12, 20\}α∈{4,8,12,20}

  and quantified trade-offs between faster estimation dynamics vs. noise sensitivity and control effort.

LQG output feedback (system-level control reasoning):

- Built LQG as **LQR + continuous-time Kalman filter**, leveraging the separation principle.
- Used measurement noise settings (example best-run settings):

  σx=0.02\sigma_x = 0.02σx=0.02 m, σθ1=1∘\sigma_{\theta_1} = 1^\circσθ1=1∘, σθ2=1∘\sigma_{\theta_2} = 1^\circσθ2=1∘ (converted to radians for covariance consistency).
- Compared behavior on both the **linearized** model and the **original nonlinear** model to validate robustness beyond local linear assumptions.

Constant disturbance rejection via integral augmentation (LQI):

- Addressed constant force disturbance on the cart by augmenting an integral state

  z(t)=∫(x−xref)dtz(t) = \int (x - x_{ref}) dtz(t)=∫(x−xref)dt

  and designing an augmented controller u=−Kx^−Kizu = -K\hat{x} - K_i zu=−Kx^−Kiz.
- Explicitly noted the required feasibility checks (augmented controllability/stabilizability and estimator detectability).

Resume bullets (pick 2–4)

- Derived a nonlinear cart–double-pendulum crane model via Euler–Lagrange mechanics and implemented nonlinear state-space simulation for control evaluation.
- Linearized the plant at equilibrium and designed LQR feedback with Q=diag(1,0.1,10,0.5,10,0.5)Q=\mathrm{diag}(1,0.1,10,0.5,10,0.5)Q=diag(1,0.1,10,0.5,10,0.5), R=0.01R=0.01R=0.01; verified stability via eigenvalue analysis.
- Benchmarked Luenberger observers across multiple measurement outputs using pole-placement speed factors α∈{4,8,12,20}\alpha \in \{4,8,12,20\}α∈{4,8,12,20}, quantifying ISE vs. peak control effort trade-offs.
- Implemented LQG output feedback (LQR + Kalman filter) and extended to LQI via integral augmentation to reject constant force disturbances and remove

steady-state tracking offsets.

Keywords

nonlinear dynamics, Euler–Lagrange, linearization, controllability/detectability, LQR, Luenberger observer, pole placement, Kalman filter, LQG/LQI, disturbance rejection, simulation-based validation

---

4B-3) ENPM667 Project 1 — Follow-up Support: Time-Varying Stiffness Modeling + CFC vs FLC Control

Repo: https://github.com/jerry102102102/ENPM667_Project1

Artifact: ENPM667_project_1.pdf

Time: Nov 2025

One-liner (resume-ready)

Reproduced and evaluated a follow-up support control framework for thin-plate machining, modeling time-varying local stiffness and comparing CFC vs. FLC controllers under matched simulation conditions with quantified accuracy–effort trade-offs.

What the project is (professional explanation)

Thin-walled workpieces deform easily during machining. A **follow-up support head** (robot-carried, gas-spring modules) tracks the cutter and provides **localized, time-varying normal stiffness** only where needed. The core technical challenge is that the effective stiffness is **position-dependent and time-varying**, coupling mechanics, actuation (pressure), and control.

Deep technical notes (detail bank)

Modeling scope (what you actually modeled):

- Local workpiece stiffness kw(xw,yw)k_w(x_w, y_w)kw(xw,yw) varies with contact position along the tool path.
- Pneumatic spring stiffness ks(p,l)k_s(p, l)ks(p,l) varies with pressure/extension.

- Robot end-effector normal stiffness kr(θ)k_r(\theta)kr(θ) contributes additional compliance.
- Combined effects create a time-varying closed-loop stiffness environment that must be controlled.

Fair, apples-to-apples controller comparison setup (strong engineering habit):

- Compared both controllers on the same plant and time window:

  t∈[0,3π]t \in [0, 3\pi]t∈[0,3π], Δt=10−3\Delta t = 10^{-3}Δt=10−3 s
- Used the same sinusoidal reference:

  z¯w(t)=10−3sint\bar{z}_w(t) = 10^{-3} \sin tz¯w(t)=10−3sint
- Enforced the same pressure saturation: ±1 MPa

Controller parameterization (concrete evidence you really ran it):

- Best working values used in the comparison run:
  - **CFC:** κ1=150\kappa_1 = 150κ1=150, κ2=1×108\kappa_2 = 1 \times 10^8κ2=1×108 (with inner P gain = 100)
  - **FLC:** kp=3.6×103k_p = 3.6 \times 10^3kp=3.6×103, kd=110k_d = 110kd=110

Quantified trade-off result (accuracy vs effort):

- Aggregated metrics from matched runs yielded:
  - **Error ratio (CFC : FLC) ≈ 0.88 : 1** → CFC achieved ~**12% lower** total tracking error
  - **Cost ratio (CFC : FLC) ≈ 1 : 0.98** → CFC required ~**2% higher** total control cost/effort
- Interpretation: CFC slightly increases pressure effort but reduces tracking error—consistent with the expected accuracy–effort trade-off.

Resume bullets (pick 2–3)

- Modeled time-varying stiffness in follow-up support machining (position-dependent workpiece stiffness and pressure-dependent pneumatic actuation) and implemented reproducible simulation evaluation.

- Ran side-by-side CFC vs FLC comparison under matched conditions (Δt=10−3\Delta t=10^{-3}Δt=10−3 s, same reference, ±1 MPa saturation) and quantified accuracy–effort trade-offs.
- Achieved lower tracking error with CFC (error ratio ≈ 0.88 vs FLC) at slightly higher control cost (cost ratio ≈ 1 vs 0.98), demonstrating tunable performance trade-offs via controller parameters.

Keywords

time-varying stiffness, compliance control, pneumatic actuation, machining support, controller comparison, reproducible simulation, trade-off analysis, parameter study

---

5) Skills（全量 + 可按 JD 精簡）

> 這裡是「最大集合」，你做 JD tailored resume 時再挑對位的 12–18 個放履歷即可。

5A) Programming / Languages

- Python, C/C++, SQL
- JavaScript/TypeScript (SWE track)

5B) Robotics / Controls / Embedded

- ROS2, Gazebo, MoveIt; TF, URDF
- Control systems: PID, trajectory tracking, system identification, kinematics/dynamics, state-space control (LQR/LQG/LQI), observers (Luenberger/Kalman)
- Real-time concepts (Linux userland): threads/timers, scheduling, timing/jitter profiling, QoS
- Microcontroller fundamentals (ADC/GPIO/PWM/SPI/I2C), FreeRTOS concepts (in progress)
- Lab/bench: oscilloscope, DMM, soldering/crimping, ESD discipline, PPE/safety interlocks

5C) ML / MLOps

- PyTorch; RL (DQN, PPO), scikit-learn, pandas, NumPy
- Experiment tracking / rollout workflows (MLflow in SWE track)

5D) Data / Cloud / DevOps / Backend

- Azure Event Hubs, Databricks (DLT, PySpark, Delta)
- Docker, Kubernetes (basic), Jenkins/GitHub Actions, Linux, Git
- Observability: logs/metrics/tracing; CI/CD; blue–green deployment
- Backend: Node.js (Nest.js), Express; Prisma; PostgreSQL (schema/index/query optimization)

---

6) Soft Skills / Signals（可用來「選 bullet」的判斷依據）

你這些素材其實很值錢，只是要在履歷裡用「工程語言」呈現：

- **Ownership:** owned/led a platform end-to-end; shipped production pipelines
- **Reliability mindset:** regression tests, fail-fast validation, SLO alerting, observability, rollback, watchdog/fallback
- **Cross-functional execution:** clear handoffs, API/data contracts, documentation/runbooks
- **Cost awareness:** cloud cost -20%
- **Evidence-driven:** quantified outcomes (CTR, success rate, PR-AUC, error/cost ratios)

---

7) Bullet Bank（可直接複製貼到履歷；按 JD 抽）

Robotics / Controls

- Built ROS2 + Gazebo motion pipeline: DLS IK with numerical safeguards, 2π unwrapping, and robust JointTrajectory execution via ros2_control controllers.
- Designed LQG output feedback for a nonlinear cart–double-pendulum system, validating performance on both linearized and original nonlinear dynamics.
- Added integral augmentation (LQI) for constant disturbance rejection and steady-state error elimination; verified feasibility via augmented controllability/stabilizability checks.
- Designed sim-to-real RL joint-control framework with watchdog/fallback safety patterns and timing/jitter profiling for deployment stability.

SWE / Backend / Platform

- Owned and scaled backend microservices (NestJS, Prisma, PostgreSQL); implemented schema evolution, performance tuning, and production observability.
- Built CI/CD with Docker and automated test gates; executed safe deployments via blue-green strategies with rollback readiness.

Data / Applied ML / MLOps

- Built real-time streaming ETL (Event Hubs → Databricks DLT) with idempotency + SLO alerting, supporting millions/day and ~5-min latency.
- Led enterprise BI LLM agent platform with HITL gating and regression suites; improved analytics success +35% and reduced cloud cost -20%.
- Implemented DQN-based recommender robust to drift/cold-start; improved CTR +20% and engagement +15%, with PR-AUC +0.07.

---

8) How to Tailor (操作規則：你之後每次改履歷就照這個流程)

1. 先選方向：Robotics/Controls vs SWE/Platform vs Applied ML
2. Summary 用 1A/1B 模板替換

3. Experience：STARK Tech 保留，但 bullets 只挑 **JD 最對位的 3–5 條**
4. Projects：
   - Robotics JD：優先拿 **ENPM662 motion pipeline** + **ENPM667 LQG/LQI** + **RL Brain Trainer**
   - SWE JD：優先拿 **BI agent platform** + **streaming ETL** + **microservices/CI/CD**
5. Skills：只放 JD 會過 ATS 的關鍵字（12–18 個），其他不要塞