# What if… — Dev Canvas: Tổng quan logic & hướng triển khai

**Mục đích:** tài liệu canvas này tóm tắt những quyết định logic quan trọng, mapping stat ↔ event, cấu trúc event chain, config difficulty, energy/time, fail states, meta formula, và quy tắc AI. Dùng để dev & content designer có thể triển khai code và viết content mà không mâu thuẫn.

## 1. Tóm tắt nhanh

* Mỗi **run** (=một đời) sở hữu seed để RNG **deterministic**.
* Core loop: **SelectEvent -> PresentChoices -> Resolve (checks + RNG + modifiers) -> ApplyEffects -> AdvanceTime/Energy -> Repeat**.
* Data-driven: events, choices, effects lưu JSONB (DB) / JSON files; logic engine đọc và thực thi.

## 2. Mapping Stat ↔ Event Tag (bảng mẫu)

| Stat | Ảnh hưởng chính lên event tags | Modifier ví dụ |
| --- | --- | --- |
| Luck | opportunity, randomCheck (tăng weight & successChance) | weight += 0.01 \* luck, chance += 0.005 \* luck |
| RiskLevel | accident, crime, scandal (tăng weight), stableJob (giảm weight) | weight \*= 1 + riskLevel/100 |
| Knowledge | education, research, scholarship (mở choice) | unlock choice if knowledge >= threshold |
| Energy | event chỉ xuất hiện nếu energy >= cost | block or force rest |
| Finance | events về investment, loan, career | availability + choices cost check |
| Relationship | help, referral, romance events | increase chance of social outcomes |
| Reputation | high → career\_opportunity, media events | boost weight for high-tier branches |

**Quy tắc:** Dev/Content-designer luôn tham chiếu bảng này khi tạo event mới. Nếu một event muốn target stat khác, thêm tag tương ứng và khai báo modifier rõ ràng trong event JSON.

## 3. Thang điểm & Unit stat

* **Range chung:** 0..100 cho hầu hết (HP, Mood, Energy, Knowledge, Status, Relationship)
* **Finance:** integer (có thể âm)
* **Luck / RiskLevel / Reputation:** -50..+50 hoặc 0..100 (chọn 1 kiểu và dùng nhất quán)

**Delta classes (gợi ý):**

* Tiny: ±1–5 (micro change)
* Small: ±6–15
* Medium: ±16–35
* Large: ±36–70
* Extreme: >70 (hiếm, ending-trigger)

Content-author phải gán delta theo class để giữ consistency.

## 4. Difficulty config (object mẫu)

{  
 "Easy": {"baseOddsMultiplier": 1.2, "negativeEventBias": 0.5, "initialLuck": 10},  
 "Medium": {"baseOddsMultiplier": 1.0, "negativeEventBias": 1.0, "initialLuck": 0},  
 "Hard": {"baseOddsMultiplier": 0.8, "negativeEventBias": 1.5, "initialLuck": -5},  
 "Asian": {"baseOddsMultiplier": 0.75, "negativeEventBias": 1.8, "initialDebuffRate": 0.2}  
}

* baseOddsMultiplier nhân lên baseChance khi resolve check.
* negativeEventBias tăng weight cho negative tags.
* Đặt config này thành file JSON/TS constant để dễ tuning.

## 5. Energy & Time mapping (quy tắc)

* **Energy pool per phase:** mặc định 100 energy cho mỗi life-phase (childhood / school / adult / elder) hoặc mỗi “milestone span”.
* **Event.energyCost:** mỗi event có energyCost (0, 5, 10, 25…).
* **Time progression rule (đề xuất):**
  + 1 **turn** = 1 event. Map turnsPerYear = 4 (ví dụ) ⇒ 4 events ~ 1 year. Tùy event có trường ageAdvance để tăng tuổi trực tiếp.
  + Khi energy <= 0 → force rest event (rest restores energy & mood)

Quy tắc này giúp AI/designer biết 1 event tương ứng bao nhiêu time unit.

## 6. Fail states & Ending types (mở rộng)

**Auto-end triggers** (ví dụ):

* HP <= 0 → ending: death\_health (bad)
* Finance <= -1000 (configurable) → ending: bankruptcy (special bad)
* Mood <= 0 + burnoutFlag → ending: burnout (special)
* age >= maxAge or pass retirement milestone → normal/good/poor ending depending stats

**Ghi chú:** hãy lưu list thresholds trong config file để dễ tuning.

## 7. Event chain data (model & lưu state)

**Event schema (core fields):**

* id, type, title, description, min\_age, max\_age, prerequisites, tags, branch\_weight, choices[].

**Choice schema (core fields):**

* id, text, energyCost, baseChance?, effects (deterministic), successEffects/failureEffects, chainedEventIdOnSuccess, chainedEventDelay (turns/age)

**State to store for chains:**

* flags: map<string, boolean> (e.g., is\_enrolled\_university:true)
* scheduledEvents: array of { eventId, executeAtAge?, executeAfterTurns? }
* currentPhase (childhood, school, adult)

**Usage:** khi choice trigger chain -> push to scheduledEvents with executeAfterTurns or executeAtAge.

## 8. Meta points formula & shop balance

**Sample formula:**

survivalPoints = age \* 10  
eventPoints = eventsCount \* 5  
goalPoints = sum(miniGoal.points)  
endingMultiplier = {good:1.5, normal:1.0, bad:0.6}  
runScore = floor((survivalPoints + eventPoints + goalPoints) \* endingMultiplier)

**Tuning:** xác định maxRunScore giả định (ví dụ 1000) rồi design price list talents/perks dựa trên tỷ lệ: cheap(50), medium(200), rare(500). Test simulate thousand runs để get average runScore.

## 9. Parametric event template (giảm số event viết tay)

**Template example:**

{  
 "templateId":"opp\_job\_offer",  
 "templateText":"{company} offers {playerName} a job as {role}.",  
 "params":["company","role","salary","location"]  
}

* Engine fills params from pool (companies list, roles list) + seed to be deterministic.
* Template + small param set yields nhiều biến thể.

## 10. Tag system (đề xuất phân tầng)

* Level 1 (domain): career, health, relationship, education, finance, art, sports, random
* Level 2 (subtags): career:job\_offer, education:scholarship, health:accident
* Level 3 (attributes): rarity:rare, impact:major

Engine có thể filter/boost bằng tags: e.g., if player has talent 'art' then boost events with tag 'art' by +0.2.

## 11. Convergence checkpoints & phases

* Đặt **checkpoint phases**: birth(0), childhood(1-6), school(7-18), young\_adult(19-30), midlife(31-60), elder(61+).
* Ở cuối mỗi phase, force a **convergence event** that many branches map back to (e.g., graduation, marriage decision, career pivot). Giúp control narrative drift.

## 12. AI sandbox rules (giai đoạn đầu)

* **Role:** chỉ generate flavor text + rephrase choices; không produce core logic (effects, prerequisites) ở giai đoạn beta.
* **Validation:** every AI output -> Zod/AJV schema validate -> if fail, retry up to N times -> fallback to scripted.
* **Cache:** store successful AI outputs in ai\_cache keyed by signature\_hash.
* **Gradual expansion:** after stable tests, allow AI produce successEffects/failureEffects but still validate with rules (max delta per class).

## 13. Developer checklist (ngắn hạn)

## 14. Gợi ý bổ sung (để hoàn thiện)

* Thêm audit\_logs table để debug run restores.
* Build small simulation harness (Node) để chạy 10k runs and collect stats for balancing.
* Document content author guidelines (delta classes, tags, template params) và provide an Excel/CSV template for event writers.

### Kết luận

Tất cả các gợi ý bạn nêu đều hợp lý và phù hợp với hướng scale mà dự án cần. Mình đã gom lại, chuẩn hóa và thêm các mẫu config/schema để dev team bắt tay làm ngay.

Nếu bạn đồng ý, mình sẽ:

1. Xuất bản file này lên canvas (đã xong) — team có thể xem/edit.
2. Tiếp theo generate: (A) shared/types.ts + Zod schemas cho Event/Choice/Effect, (B) một file difficulty.config.json, (C) sample parametric-templates.json, và (D) simulation script để test balancing.

Bạn muốn mình làm bước (A/B/C/D) nào trước? Hoặc cần chỉnh gì trong canvas này không?