# What if… — Game Design Document (GDD) + Flowchart

**Version:** 0.1 **Author:** Drafted with ChatGPT + You **Updated:** 2025-08-12

## Table of contents

* [Overview](#overview)
* [Core Loop](#core-loop)
* [Meta Progression](#meta-progression)
* [Stats & Mechanics](#stats--mechanics)
* [Event System](#event-system)
* [RNG, Difficulty & Branch Weights](#rng-difficulty--branch-weights)
* [AI Integration Strategy](#ai-integration-strategy)
* [UI / UX Layout](#ui--ux-layout)
* [Data Models & Schemas](#data-models--schemas)
* [Flowchart (Gameplay)](#flowchart-gameplay)
* [Roadmap & Milestones](#roadmap--milestones)
* [Appendix: Prompt Samples & Notes](#appendix-prompt-samples--notes)

## Overview

**Title (temp):** *What if…*

**Genre:** Narrative Choice-Based + Roguelite elements (non-turn-based, training-game rhythm)

**Platform (MVP):** Web (PC-first). Mobile later.

**High-level concept:** Play multiple “lives”. Each life is a sequence of events → choices → consequences. RNG + AI provide variation. Each run yields meta-points used to unlock talents/perks that affect future runs.

## Core Loop

Anchor: #core-loop

1. **Receive Event**: Engine picks next event (scripted or AI-generated) given profile & state.
2. **Present Choices**: 2–4 options shown to player.
3. **Resolve Choice**: Engine computes result using stat checks + RNG + modifiers.
4. **Apply Outcome**: Update stats, trigger chained events, record timeline entry.
5. **Advance Time**: Progress within the current life (age/milestone). If life-end -> compute ending and meta points.
6. **Meta Progression** (between runs): Spend points on talents/perks; start new run.

## Meta Progression

Anchor: #meta-progression

* **Points (Run Score):** Calculated from: survival time, events triggered, mini-goals completed, ending quality.
* **Spendable items:** Talents (art, music, sports…), Perks (Luck Boost, Risk Reducer), Stat caps, Unlock new event pools.
* **Purpose:** Provide long-term progression and give players sense of growth across runs.

## Stats & Mechanics

Anchor: #stats--mechanics

**Visible stats**

* HP (Health)
* Mood
* Finance
* Status (social)
* Relationship
* Energy (per milestone, e.g., 100 energy/milestone)

**Hidden / Advanced stats**

* Luck
* Knowledge / Experience
* Reputation
* Risk Level (hidden)
* Karma (optional)

**Mechanics**

* **Energy model:** Each event has an energy cost (0 or >0). Rest actions recover energy and modify Mood.
* **Skill / Talent checks:** Some choices require minimum Knowledge or CareerSkill; success chance modified by Luck.
* **Perks/Talents:** Persist across runs once unlocked.
* **Mini-goals:** Short-term objectives (e.g., graduate by age X) that affect run score and branching.

## Event System

Anchor: #event-system

**Types of events**

* **Scripted (canonical)**: Pre-authored events critical to narrative (e.g., schooling milestones).
* **Random (procedural)**: RNG-driven occurrences (e.g., accidents, chance encounters).
* **AI-Generated**: Parameterized events created by LLMs using templates + context.

**Event structure (concept)**

* id
* type (milestone, random, opportunity, chain)
* min\_age / max\_age
* prerequisites (stats, history flags)
* description (text)
* choices[] (each with effects[] and possible chained\_event\_id)
* tags, rarity, branch\_weight

**Event chaining**

* Events may set flags that unlock or bias future events.
* Chain examples: Learn instrument -> local band -> performance -> fame branch.

## RNG, Difficulty & Branch Weights

Anchor: #rng-difficulty--branch-weights

**Difficulty modes** (affect base odds and debuffs)

* Easy: 95/5
* Medium: 75/25
* Hard: 50/50 (or 75/25 with harsher debuffs depending on design)
* Special (e.g., “Asian” profile): custom distribution, increased early-life debuffs

**Modifiers**

* Luck, Relationship, Skill add/subtract to outcome odds.
* Branch weights control rarity of event branches (e.g., 1% celebrity branch).

**Design guidelines**

* Never allow pure-random events to break narrative sense (prerequisites must hold).
* Use convergence nodes (many branches rejoin) to limit combinatorial explosion.

## AI Integration Strategy

Anchor: #ai-integration-strategy

**Role for AI**

* Generate event descriptions & dialogues from templates + context
* Propose variable values/effects based on character state
* Rewrite choice text for tone consistency

**Safety & Control**

* All AI output must match a strict JSON schema. If validator fails -> fallback to curated event.
* Moderation/filter layer to remove harmful content.

**Cost-saving hybrid approach**

* **Pre-generate** large pools of AI events offline and cache them.
* Use **local LLMs** or free-tier APIs during development (e.g., Hugging Face, small LLaMA variants).
* Reserve premium API calls for special features or curated sequences.

**Prompting pattern**

* Provide: character\_profile, current\_state, recent\_events, event\_template, seed
* Require: JSON-only response following schema examples.

## UI / UX Layout

Anchor: #ui--ux-layout

**PC-first layout**

* Left: Narrative panel (title + description + image placeholder)
* Right: Stat panel (HP, Mood, Finance, Energy, Luck, Relationship)
* Bottom: Choice buttons (2–4) with short descriptions
* Top bar: Age/Time, Quick actions (Save, Menu, What-if rewind)
* Event Log/Timeline: Collapsible sidebar or modal to review past choices

**UX notes**

* Color tone shifts slightly by Mood.
* Minimal animations. Focus on readability.
* 3 save slots by default; autosave after each milestone.

## Data Models & Schemas

Anchor: #data-models--schemas

**Event JSON (example)**

{  
 "id": "evt\_00123",  
 "type": "school\_opportunity",  
 "title": "Học bổng trung học",  
 "description": "Bạn được thầy giới thiệu tham gia kỳ thi học bổng...",  
 "min\_age": 13,  
 "choices": [  
 {"id":"c1","text":"Tham gia","effects":[{"stat":"education","delta":2}]},  
 {"id":"c2","text":"Từ chối","effects":[{"stat":"social","delta":1}]}  
 ],  
 "tags": ["education","early\_life"],  
 "branch\_weight": 0.8  
}

**Player Save (concept)**

{  
 "player\_id":"...",  
 "seed": 12345,  
 "profile": {"background":"normal","traits":[]},  
 "stats": {"hp":80,"mood":60,...},  
 "timeline": [{"age":10,"event\_id":"evt\_00123","choice":"c1"}, ...]  
}

## Flowchart (Gameplay)

Anchor: #flowchart-gameplay

START (New Run)  
 |  
[Generate Profile & Seed]  
 |  
[Select Next Event]  
 |--(Check prerequisites: age, tags, stats?)--> if NO -> pick next  
 |  
[Present Event (text + choices)]  
 |  
[Player Selects Choice]  
 |  
[Resolve Choice]  
 | - compute base odds (difficulty)  
 | - apply stat modifiers (luck, skill)  
 | - RNG roll  
 | - AI may rewrite text / compute additional effects  
 |  
[Apply Effects] -> update stats, timeline, set flags  
 |  
[Advance Time / Energy decrement]  
 |  
{Is Life End?} --No--> loop back to [Select Next Event]  
 --Yes--> [Compute Ending & Run Score] -> [Grant Meta Points]  
 -> [Player chooses Meta purchases?] -> [New Run or Exit]

**Notes:**

* Event chaining: choices may enqueue future events or set flags.
* Convergence nodes: multiple paths can lead to same canonical event to control combinatorial growth.

## Roadmap & Milestones

Anchor: #roadmap--milestones

**Phase 0 – Prep (1 week)**

* Finalize GDD (this doc)
* Create 20–30 seeded events (canonical)
* Design event schema + DB model

**Phase 1 – MVP (4–6 weeks)**

* Backend: state engine + API endpoints (next-event, apply-choice, save/load)
* Frontend: basic UI (narrative panel, stats, choices)
* Integrate 30 events, deterministic RNG via seed

**Phase 2 – AI & Scale (6–8 weeks)**

* Implement AI integration + validator
* Pre-generate event pools and caching
* Add meta-progression (points, talents, perks)

**Phase 3 – Beta & Polish (4–6 weeks)**

* Open Beta deployment, analytics, feedback loops
* UI polish, accessibility, extra content

## Appendix: Prompt Samples & Notes

Anchor: #appendix-prompt-samples--notes

**Prompt template (short):**

System: You are a narrative event generator. Output ONLY JSON matching the schema.  
Input: character\_profile={...}, current\_state={...}, seed=123  
Produce: {id, type, title, description, choices[]}

**Moderation:** Always run AI output through a sanitizer and the JSON schema validator.

**Playtesting:** Instrument decision heatmaps and event frequency to rebalance branch weights.

## Next steps (recommended right now)

1. Import this doc into project repo (README / docs folder).
2. Create a small dataset of 30 canonical events (JSON files) and wire MVP engine to consume them.
3. Build frontend skeleton and simple backend loop.
4. Run a closed test with 5–10 players, collect metrics.

*End of document.*