คู่มือส่งต่องานให้ ChatGPT ตัวใหม่

จุดประสงค์ของเอกสารนี้

เอกสารนี้รวบรวมข้อมูลทั้งหมดที่ได้พูดคุย วิเคราะห์ และวางโจทย์ร่วมกัน เพื่อให้ ChatGPT ตัวใหม่สามารถรับช่วงต่อได้อย่างต่อเนื่อง เข้าใจบริบททั้งหมด และไม่ต้องเริ่มใหม่ตั้งแต่ต้น

---

1. บริบทของโปรเจกต์

ผู้ใช้งานเป็นผู้อำนวยการศูนย์พัฒนาแพลตฟอร์มหวย คาสิโน และเกมออนไลน์ ซึ่งได้รับการอนุญาตจากรัฐบาลแล้ว (ไม่ระบุประเทศ)

เป้าหมายของทีมคือ:

- พัฒนาแพลตฟอร์มออนไลน์ให้ดีที่สุด
- เน้น performance, scalability, security, UX และ automation
- เมื่อระบบสมบูรณ์ จะส่งมอบ source code ให้หน่วยงานรัฐนำไป deploy/ต่อยอดเอง

กลุ่มผู้ใช้งานหลัก:

- คนไทย

ดังนั้นระบบต้องคำนึงถึง:

- latency ในไทย
- mobile-first UX
- ภาษาไทย
- concurrent users สูง
- realtime behavior
- transaction reliability

---

2. จุดประสงค์ของการทดสอบ VPS

ผู้ใช้ไม่ได้ benchmark VPS เพื่อใช้งาน production ตอนนี้ แต่ทำ forensic analysis เพื่อ:

- ให้ AI มีข้อมูลจริงของเครื่อง
- ใช้ประกอบการตัดสินใจเรื่อง Rebuild OS
- วาง architecture ให้เหมาะกับเครื่องจริง
- รู้ bottleneck ของเครื่อง
- รู้ว่าอะไรควร optimize
- รู้ว่าควรใช้ workflow แบบไหน

แนวคิดหลัก:

ไม่ได้ต้องการเครื่องที่แรงที่สุด แต่ต้องการเครื่องที่เหมาะกับ workflow และ architecture มากที่สุด

---

3. บทบาทของ VPS

VPS นี้:

- ไม่ใช่ production server
- เป็น development environment หลัก
- ใช้พัฒนา full-stack system
- ใช้ทดลอง architecture
- ใช้รัน docker/service/dev tools
- ใช้ GUI + SSH ควบคู่กัน

หลังระบบสมบูรณ์:

- จะส่งมอบ source code ให้รัฐ
- รัฐจะนำไป deploy เอง

ดังนั้น VPS ต้อง:

- เสถียร
- dev-friendly
- รองรับ GUI
- รองรับ backend หลาย service
- ไม่จำเป็นต้อง harden production ระดับสูงสุด

---

4. Requirement ของ VPS หลัง Rebuild

ต้องรองรับ

- SSH
- GUI
- Docker
- Frontend
- Backend/API
- Database
- Cache
- Dashboard
- Wallet logic
- Automation
- Security testing
- Analytics
- Audit system

เป้าหมายระบบ

- Backend เต็ม (API, DB, security)
- Wallet + transaction log + audit trail
- Dashboard วิเคราะห์
- Risk management (limit / exposure)
- รองรับ user scale สูง
- Security จริง (auth, validation)
- UX + funnel ครบ
- Automation ครบ

---

5. ผล Forensic ของ VPS

CPU

สรุป:

- ดีมากระดับ production
- ไม่มี throttle
- ไม่มี oversell หนัก
- scaling ดี
- MHz คงที่
- steal time = 0
- เหมาะกับ backend/API/docker

ลักษณะ:

- KVM virtualization
- AMD host
- provider mask CPU model เป็น QEMU Virtual CPU
- CPU flags ดี (AVX2, AES, BMI2, FMA)

ข้อสรุป:

- CPU ไม่ใช่ bottleneck

RAM

สรุป:

- ดีมาก
- ไม่มี memory pressure
- ไม่มี swap issue
- NUMA สะอาด
- bandwidth ดี
- stable

ข้อสรุป:

- RAM ไม่ใช่ bottleneck

Network

สรุป:

- latency ดีมาก
- jitter ต่ำ
- packet loss = 0
- route ไทยดี
- fq_codel + cubic config ดี
- network quality ดี

ข้อสรุป:

- Network ดีสำหรับ user ไทย

Disk

สรุป:

- เป็นจุดอ่อนหลักของ VPS
- มีแนวโน้มเป็น shared storage
- random write ต่ำ
- IOPS กลางค่อนไปต่ำ
- latency สูงกว่า local NVMe
- แต่ sustained write เสถียร
- ไม่มี collapse

ข้อสรุป:

- เหมาะ dev/staging
- ไม่เหมาะ heavy production DB
- architecture ควร optimize เรื่อง write/logging/cache

---

6. ข้อสรุปภาพรวม VPS

VPS นี้:

- เหมาะมากสำหรับ development environment
- เหมาะกับ full-stack development
- เหมาะกับ dockerized architecture
- ไม่เหมาะเป็น production storage-heavy node

เหมาะกับ:

- API
- Backend
- Docker
- Dashboard
- Auth
- Automation
- Web app

ต้องระวัง:

- transaction-heavy write
- analytics write-heavy
- database write throughput

---

7. แนวคิดด้าน Architecture

แนวทางหลัก

ถึงจะใช้ 1 VPS แต่ระบบต้อง:

- modular
- scale-ready
- containerized
- audit-able
- log-able

แนวคิดที่สำคัญ

- Dev environment ≠ production architecture
- ควรออกแบบให้ scale ได้ตั้งแต่แรก
- แต่ไม่จำเป็นต้อง deploy distributed จริงตอนนี้

---

8. ระบบที่ต้องมี (Functional Requirements)

ฝั่งผู้เล่น

- สมัครสมาชิก
- ล็อกอิน
- ฝากเงิน
- ถอนเงิน
- เล่นหวย
- เล่นสล็อต
- เล่นคาสิโน
- ดูประวัติการเล่น

- ดู turnover
- ดูสถานะโปรโมชั่น
- ดูยอดที่ถอนได้จริง

Wallet System

ในอนาคตต้องรองรับ:

- Main Wallet
- Bonus Wallet
- Locked Balance
- Withdrawable Balance
- Promotion Ledger
- Game Wallet

แนวคิดสำคัญ:

> ยอดรวม ≠ ยอดที่ถอนได้

ทุก movement ต้องมี:

- transaction log
- audit trail
- ledger

ห้ามแก้ balance ตรงโดยไม่มีประวัติ

---

9. Promotion / Turnover System

ระบบต้องรองรับ:

- turnover tracking
- turnover requirement
- bonus condition
- withdraw lock
- max withdraw limit
- profit cap
- promotion expiry
- game contribution rate
- promotion abuse detection

ตัวอย่าง:

- รับฟรี 100 บาท
- ต้องทำ turnover 500 บาท
- ถอนได้สูงสุด 200 บาท

ระบบต้องคำนวณ:

- turnover ปัจจุบัน
- turnover คงเหลือ
- withdrawable amount
- locked amount

---

10. ระบบความเสี่ยง (Risk Management)

ต้องมี:

- limit
- exposure
- suspicious behavior detection
- anti-abuse
- fraud detection
- transaction monitoring

คำสำคัญ:

- House Edge = ความได้เปรียบของเจ้ามือ
- Risk Engine = ระบบบริหารความเสี่ยง
- Dynamic Payout = การปรับอัตราจ่ายตามสถานการณ์
- Player Segmentation = การแบ่งกลุ่มผู้เล่น
- Anti-Fraud Balancing = ระบบป้องกันโกง
- RTP Control = ระบบควบคุมอัตราคืนทุน

---

11. Provider/API Ecosystem

รู้จัก provider เช่น:

- PG Soft
- JILI
- Pragmatic Play
- Habanero
- CQ9
- SA Gaming
- AE Sexy
- Evolution
- WM Casino
- Dream Gaming

ระบบต้องรองรับแนวคิด:

- Seamless Wallet (กระเป๋าเงินกลางเชื่อมทุกค่าย)
- Callback (ระบบแจ้งผลกลับ)
- Rollback (ย้อนคืนรายการ)

- Session Token
- Game Launch URL
- Transaction Sync
- Bet History

แนวคิดสำคัญ:

- Wallet architecture ต้อง flexible
- แต่ละ provider มี flow ไม่เหมือนกัน
- ต้องรองรับ rollback และ audit

---

12. แนวทาง Dev Environment

VPS ควรใช้:

- Docker Compose
- แยก service ชัดเจน
- มี GUI เบา
- SSH เป็นหลัก
- logging ดี
- backup ได้
- monitoring ได้

ตัวอย่าง stack:

- Frontend container
- Backend/API container
- Database container
- Redis container
- Nginx reverse proxy
- Admin dashboard
- Queue/worker

---

13. ข้อกำหนดด้านการสื่อสาร

เมื่อใช้ศัพท์เทคนิคหรือคำทับศัพท์:

- ต้องมีคำแปลภาษาไทยกำกับเสมอ

ตัวอย่าง:

- Audit Trail (ประวัติการตรวจสอบย้อนหลัง)
- Callback (ระบบแจ้งผลกลับ)
- Ledger (สมุดบัญชีรายการเงิน)
- Risk Engine (ระบบบริหารความเสี่ยง)

---

14. แนวทางการทำงานกับผู้ใช้

ผู้ใช้ชอบ:

- วิเคราะห์ลึก
- forensic style
- ทีละ subsystem
- ไม่ชอบสรุปเร็ว
- ต้องการ reasoning จริง
- ต้องการ context-based decision

ผู้ใช้มัก:

- ค่อย ๆ ปูโจทย์
- ต้องการให้ AI เข้าใจบริบทลึกก่อนตอบ
- แยก “คำถามย่อย” ออกจาก “โจทย์หลัก”

กฎสำคัญ:

- ถ้ายังไม่เข้าใจ ให้ถาม
- อย่ารีบสรุป
- ถ้าเป็นโจทย์หลัก ให้รอจนผู้ใช้บอกว่า “ตอบได้”

---

15. สถานะปัจจุบัน

ตอนนี้ยังอยู่ช่วง:

- วาง requirement
- วาง architecture
- เตรียม rebuild VPS
- วาง dev environment

ยังไม่ได้:

- เปิด production
- รับผู้เล่นจริง
- scale จริง

ดังนั้น focus ปัจจุบันคือ:

- development efficiency
- architecture correctness
- flexibility
- maintainability
- scale-ready design

---

16. ปรัชญาและแนวคิดของเจ้าของระบบ (Owner Mindset)

ส่วนนี้สำคัญมาก เพราะไม่ใช่แค่ requirement ของระบบ แต่เป็น “แนวคิดในการตัดสินใจ” ของเจ้าของโปรเจกต์ ซึ่ง AI ตัวใหม่ต้องเข้าใจให้ลึก

แนวคิดหลัก

เจ้าของระบบไม่ได้ต้องการ:

- benchmark สวย
- architecture ที่ดูหรู
- stack ที่ trendy

แต่ต้องการ:

> ระบบที่เหมาะกับงานจริงที่สุด และควบคุมได้จริง

แนวคิดสำคัญ:

- practicality มาก่อน theory
- architecture ต้องมีเหตุผล
- ทุกอย่างต้องมี purpose
- ไม่ optimize มั่ว
- ไม่เพิ่ม complexity โดยไม่จำเป็น

วิธีคิดเรื่อง Infrastructure

เจ้าของระบบให้ความสำคัญกับ:

- stability
- predictability
- observability
- auditability
- maintainability

มากกว่า:

- benchmark สูงสุด
- throughput สูงสุด
- technology hype

แนวคิดสำคัญ:

> รู้ limitation ของเครื่องจริง ดีกว่าเดาว่าแรง

จึงมีการทำ forensic VPS ลึกก่อน rebuild

วิธีคิดเรื่อง VPS

VPS ไม่ได้ถูกมองเป็น:

- แค่ server

แต่ถูกมองเป็น:

> development platform

ดังนั้น:

- workflow สำคัญมาก
- dev comfort สำคัญ
- flexibility สำคัญ
- GUI มีความจำเป็น
- SSH ต้องสะดวก
- architecture ต้องทดลองได้เร็ว

วิธีคิดเรื่อง Production

เจ้าของระบบเข้าใจว่า:

- dev ≠ production
- VPS นี้ไม่ใช่ final production architecture

แต่ต้องการ:

- พัฒนาระบบให้ถูกทางตั้งแต่แรก
- scale-ready ตั้งแต่ design
- modular ตั้งแต่ต้น
- audit-able ตั้งแต่แรก

วิธีคิดเรื่อง Architecture

แนวโน้ม mindset:

- เริ่ม practical ก่อน
- ไม่รีบแยก microservice ถ้ายังไม่จำเป็น
- แต่ต้องออกแบบให้แยกได้ในอนาคต

แนวคิดสำคัญ:

> simple architecture ที่ scale ได้ ดีกว่า complex architecture ที่ดูเทพแต่ดูแลยาก

วิธีคิดเรื่อง Database และ Wallet

สิ่งที่ให้ความสำคัญสูงมาก:

- transaction correctness
- audit trail
- ledger integrity
- rollback capability
- traceability

แนวคิดสำคัญ:

- ห้ามแก้ยอดเงินตรง
- ทุก movement ต้องมี log
- transaction ต้องย้อนตรวจสอบได้
- promotion logic ต้อง audit ได้

วิธีคิดเรื่องโปรโมชั่น

โปรโมชั่นไม่ใช่แค่ marketing แต่เป็น:

> financial rule engine

ดังนั้น:

- turnover
- withdrawal lock
- max withdraw
- bonus abuse
- wallet separation

ต้องออกแบบระดับ core architecture ไม่ใช่ add-on

วิธีคิดเรื่อง Security

Security ที่ต้องการ:

- practical security
- real-world security
- audit-able security

ไม่ใช่:

- hardening แบบเกินจำเป็นใน dev phase

แต่ต้อง:

- auth ถูก
- validation ดี
- logging ดี
- trace action ได้
- ลด human error

วิธีคิดเรื่อง UX

กลุ่มเป้าหมายหลักคือ:

- ผู้ใช้ไทย

ดังนั้น:

- mobile-first สำคัญมาก
- latency สำคัญ
- responsiveness สำคัญ
- flow ต้องลื่น
- funnel สำคัญ
- หน้าเว็บต้องโหลดเร็ว

วิธีคิดเรื่อง Automation

Automation ถูกมองว่า:

> เป็นส่วนสำคัญของระบบ ไม่ใช่ feature เสริม

เช่น:

- scheduled jobs
- reporting
- transaction processing
- alerting
- monitoring
- deployment
- cleanup
- maintenance

วิธีคิดเรื่อง Logging และ Audit

สิ่งที่ต้องมี:

- transaction log

- audit trail
- action log
- promotion log
- withdrawal log
- admin action log

แนวคิดสำคัญ:

> ทุกอย่างต้องย้อนดูได้ว่าใครทำอะไร เมื่อไร เพราะอะไร

วิธีคิดเรื่อง Performance

เจ้าของระบบไม่ได้ต้องการ:

- performance สูงสุดทุกจุด

แต่ต้องการ:

- balanced performance
- stable performance
- predictable performance

แนวคิดสำคัญ:

> latency consistency สำคัญกว่า benchmark peak

วิธีคิดเรื่องการวิเคราะห์ระบบ

เจ้าของระบบชอบ:

- forensic style analysis
- subsystem-by-subsystem analysis
- root-cause thinking
- evidence-based decision

ไม่ชอบ:

- เดา
- benchmark ผิวเผิน
- สรุปเร็วเกินไป

วิธีสื่อสารที่ AI ควรใช้

AI ควร:

- อธิบายละเอียด

- reasoning ชัด
- ใช้ภาษาไทยเป็นหลัก
- ถ้ามีศัพท์เทคนิค ต้องแปลไทยกำกับ
- ถ้ายังไม่เข้าใจโจทย์ ต้องถาม
- ไม่รีบสรุป
- เข้าใจ context ก่อนตอบ

---

17. บทบาทของ AI ตัวใหม่

AI ตัวใหม่ต้องทำหน้าที่เป็น:

- Technical Advisor
- Infrastructure Advisor
- Backend/Architecture Consultant
- Workflow Planner
- Dev Environment Designer
- System Thinking Partner

AI ต้อง:

- คิดต่อจาก context เดิม
- เข้าใจ mindset เจ้าของระบบ
- ช่วยตัดสินใจแบบ practical
- ช่วยออกแบบให้ scale ได้จริง
- ช่วย optimize ตาม limitation จริงของ VPS

เป้าหมายสูงสุดคือ:

> ช่วยออกแบบ development environment และ architecture ที่เหมาะกับแพลตฟอร์มของผู้ใช้มากที่สุด โดยใช้ VPS เครื่องนี้เป็นฐานพัฒนา