

ส่วนที่ 1

ช่องกรอกชื่อโรงแรม

เนื้อหาความ prompt

ชื่อโรงแรม: [พิมพ์ชื่อโรงแรมตรงนี้]

สโลแกนใต้โลโก้ (ไม่บังคับ): [เว้นว่าง = ใช้ STAY A LITTLE CLOSER เหมือนเดิม]

อธิบาย

วางข้อมูลที่ต้องกรอกไว้บนสุด คนกรอกจะได้ไม่ต้องหาในข้อความยาว ๆ

ใช้วงเล็บเหลี่ยม [] เพื่อให้เห็นชัดว่าตรงไหนต้องแก้ ถ้าลืมกรอก AI จะเห็นว่ายังเป็นข้อความตัวอย่างอยู่

ตอนส่งจริง ให้ใช้เฉพาะข้อความในกรอบ "เนื้อหาความ prompt" ของทั้ง 6 ส่วน ไม่ต้องใส่คำอธิบายไปด้วย

ส่วนที่ 2

บทบาทและบริบท

เนื้อหาความ prompt

คุณทำงานในฐานะ Software Engineer, Software Architect, UX/UI Designer และ AI Engineer ในโปรเจกต์เว็บบริการแขกโรงแรม (Node.js, Express, SQLite, หน้าเว็บอยู่ในโฟลเดอร์ public/) ตอนนี้ใช้ชื่อตัวอย่างว่า SALA

แขกสแกน QR ประจำห้อง ยืนยันตัวด้วยรหัสเข้าพัก แล้วส่งข้อความได้ทุกภาษา Local LLM จะแปลเป็นไทยให้พนักงาน พนักงานตอบเป็นไทย แล้วระบบแปลกลับเป็นภาษาแขก

อ่านไฟล์ AGENTS.md, README.md และ docs/ROADMAP.md ก่อนเริ่มงาน

อธิบาย

AI ไม่รู้อะไรเกี่ยวกับโปรเจกต์มาก่อน ต้องบอกว่าให้รับบทอะไร ระบบทำอะไร และให้อ่านไฟล์ไหนก่อน ถ้าไม่บอก มันจะเดาเอง และมักเดาผิด

ส่วนที่ 3

ข้อห้าม

เนื้อหาความ prompt

- ต้องคงโหมด sandbox ห้ามเปิดชำระเงินจริง ห้ามเปิดให้คนนอกโรงแรมเข้าถึง และห้ามใช้ AI บน cloud
- โมเดลที่ใช้ทดสอบตอนนี้เป็นตัวชั่วคราว ไม่ใช่ตัวที่จะใช้จริง ต้องเปลี่ยนตัวได้ภายหลัง
- งานใน docs/ROADMAP.md ห้ามทำจนกว่าเจ้าของงานจะยืนยัน ให้ทำเฉพาะงานที่สั่งใน prompt นี้
- ก่อนแก้ไฟล์ ให้สำรองไฟล์ไว้ก่อน เพราะโปรเจกต์นี้ยังไม่ได้ใช้ git

อธิบาย

AI มักขยับเกินขอบเขต เห็นอะไรก็อยากแก้ ถ้าไม่เขียนข้อห้ามไว้ มันอาจไปแตะระบบเงินหรือทำงานที่ยังไม่ได้อนุมัติ

ส่วนนี้เป็นเหมือนรั้วกันขอบเขตงาน

ส่วนที่ 4

สิ่งที่ทำไปแล้ว

เนื้อหาความ prompt

มีระบบ Recheck การแปลแล้ว:

- ข้อความที่พนักงานส่ง พอแปลแล้วจะถูกแปลย้อนกลับเป็นไทยอีกรอบ แล้วเทียบกับต้นฉบับว่า ตัวเลข คำปฏิเสธ และเจตนายังตรงกันไหม
- ข้อความที่แขกส่ง ระบบตรวจว่าจำนวนที่แปลมาตรงกับต้นฉบับ
- ข้อความเรื่องการจ่ายเงิน ที่พักอาหาร และเหตุฉุกเฉิน ระบบกักไว้ให้พนักงานตรวจทุกครั้ง
- โค้ดอยู่ใน src/recheck.js และ src/ai.js มีเทสต์ผ่านครบ 56 ข้อ ห้ามทำให้ส่วนนี้พัง

อธิบาย

บอกงานเก่าไว้ AI จะได้ไม่เขียนทับของที่ใช้ได้แล้ว

การระบุจำนวนเทสต์ (56 ข้อ) ทำให้ตรวจได้ว่าแก้แล้วยังผ่านเก่าเดิมหรือเปล่า

ส่วนที่ 5

งานที่ส่ง (เปลี่ยนชื่อ SALA)

เนื้อหาความ prompt

ถ้าช่องชื่อโรงแรมยังเป็น [พิมพ์ชื่อโรงแรมตรงนี้] ให้หยุดแล้วถามชื่อก่อน ห้ามเดาชื่อ

1. เปลี่ยนชื่อที่ผู้ใช้เห็นให้ตรงกับที่กรอกไว้ทุกตัวอักษร:

- public/index.html: ชื่อแท็บเบราว์เซอร์ (title) และคำอธิบายหน้าเว็บ (description)
- public/app.js: ชื่อโลโก้ คำว่า "SALA OPERATIONS" ในหน้าเข้าสู่ระบบพนักงาน และสโลแกน (เปลี่ยนเฉพาะเมื่อกรอกไว้)
- src/db.js: ค่าเริ่มต้น hotelName
- src/server.js: ข้อความที่แสดงตอนเปิด server

2. ฐานข้อมูลเดิมยังเก็บชื่อ SALA อยู่ ให้เปลี่ยนที่หน้าผู้ดูแล การ์ด "บริการและนโยบาย" ช่อง "ชื่อโรงแรม"

3. ชื่อภายในของโค้ด เช่น cookie (sala_session) และค่าภาษาที่เก็บในเบราว์เซอร์ (sala-language) จะเปลี่ยนก็ได้ แต่ต้องแจ้งก่อนว่าคนที่ล็อกอินอยู่จะหลุดหนึ่งครั้ง ถ้าไม่แน่ใจให้คงไว้

4. แก้ชื่อในเอกสาร README.md และ docs/ ได้ แต่ห้ามแก้ไขไฟล์ทดสอบเก่า (.json และ .txt) เพราะเป็นหลักฐาน

อธิบาย

ส่วนนี้สำคัญที่สุด ต้องบอกให้ชัดว่าแก้ไขไฟล์ไหน จุดไหน

- **ข้อ 2** คนมักสับสน เพราะแก้ไขโค้ดอย่างเดียวแล้วชื่อในฐานข้อมูลเดิมไม่เปลี่ยนตาม
- **ข้อ 3** สอนให้ AI ชั่งผลกระทบก่อนแก้ไข ไม่ใช่ค้นแล้วแทนที่คำทั้งหมดทันที
- **ข้อ 4** สอนว่าหลักฐานการทดสอบห้ามแก้ไขย้อนหลัง

ส่วนที่ 6

การตรวจและการส่งงาน

เนื้อหาความ prompt

หลังแก้ไขเสร็จ:

- ค้นหาคำว่า "sala" ทั้งโปรเจกต์ จุดที่ยังเหลือต้องมีแค่ไฟล์หลักฐานเก่า กับชื่อภายในที่ตั้งใจคงไว้
- รัน npm test ต้องผ่านครบ 56 ข้อ ถ้าไม่ผ่านให้แนบผลมาตามจริง
- เปิดเว็บดูหน้าแรกและหน้าพนักงานว่าชื่อใหม่แสดงถูกต้อง
- สรุปว่าแก้ไขไฟล์ไหนบ้าง และจุดไหนที่ตั้งใจไม่แก้ พร้อมเหตุผล
- ถ้าเจอปัญหาอื่น ให้จดเป็นข้อเสนอไว้ ห้ามแก้เอง

อธิบาย

ถ้าไม่บอกวิธีตรวจ AI มักตอบว่า "เสร็จแล้ว" โดยไม่ได้ลองจริง

ส่วนนี้บังคับให้ตรวจ 3 แบบ คือค้นคำ รันเทสต์ และเปิดหน้าเว็บดู แล้วรายงานตามจริง

สรุปหลักไว้สอน: prompt ที่ดีมี 6 ส่วน คือ ข้อมูลที่ต้องกรอก, บริบท, ข้อห้าม, งานเดิม, งานที่สั่ง, แบบระบุจุด และวิธีตรวจ ถ้าขาดส่วนไหน AI ก็จะเดาส่วนนั้นเอง

ภาคต่อ

ติดตั้งและใช้งานบนเครื่องของผู้รับงานเอง

ผู้รับงานไม่ได้ใช้ server เดียวกับเรา ต้องติดตั้งบนเครื่องตัวเอง ใช้ได้ทั้ง Mac และ Windows ทุกชั้นแยกวิธีไว้ซ้ายขวา

macOS

- Mac ชิป Apple (M1 ขึ้นไป) แนะนำ
- RAM 16 GB ขึ้นไป
- พื้นที่ว่าง 25 GB

Windows

- Windows 10 หรือ 11 (64 บิต)
- RAM 16 GB ขึ้นไป
- พื้นที่ว่าง 25 GB

ชั้น	ทำอะไร	เวลาโดยประมาณ
1	ติดตั้ง Node.js	5 นาที
2	ติดตั้ง Ollama	5 นาที
3	วางโปรเจกต์และเปิดหน้าต่างพิมพ์คำสั่ง	5 นาที
4	ดาวน์โหลดโมเดลภาษา	5–30 นาที
5	ตั้งค่าไฟล์ .env.test	3 นาที
6	สร้างบัญชีผู้ดูแลและเปิดเว็บ	5 นาที
7	ทดสอบว่าระบบทำงานจริง	10–15 นาที
8	แก้ปัญหาที่พบบ่อย	เมื่อจำเป็น

อธิบาย

ทำตามลำดับ ทำขั้น 1–7 ให้ผ่านก่อน แล้วค่อยใช้ prompt 6 ส่วนให้ AI เปลี่ยนชื่อโรงแรม เพราะ AI ต้องรันเทสต์และเปิดเว็บบนเครื่องนี้ได้

หน้าต่างพิมพ์คำสั่งคือ **Terminal** บน Mac และ **Command Prompt** บน Windows ในคู่มือนี้เรียกว่า "หน้าต่างคำสั่ง"

ขั้นที่ 1

ติดตั้ง Node.js

macOS

1. เข้าเว็บ **nodejs.org**
2. กดดาวน์โหลดตัว **LTS** สำหรับ macOS (ไฟล์ .pkg)
3. ดับเบิลคลิกไฟล์ที่ได้ กด Continue / Agree / Install จนจบ

Windows

1. เข้าเว็บ **nodejs.org**
2. กดดาวน์โหลดตัว **LTS** สำหรับ Windows (ไฟล์ .msi)
3. ดับเบิลคลิกไฟล์ที่ได้ กด Next จนจบ ช่อง "Automatically install the necessary tools" ไม่ต้องติ๊ก

macOS

เปิด Terminal: กด `⌘` + `Space` พิมพ์ **Terminal** แล้วกด Enter

Windows

เปิด Command Prompt: กดปุ่ม `Windows` พิมพ์ **cmd** แล้วกด Enter

ตรวจสอบเวอร์ชัน (เหมือนกันทั้งสองระบบ)

```
node -v
```

ต้องได้ **v24.13.0** หรือเลขที่สูงกว่า ถ้าขึ้นว่าไม่รู้จักรหัสสั่ง ให้ปิดหน้าต่างคำสั่งแล้วเปิดใหม่

อธิบาย

Node.js คือโปรแกรมที่ใช้รันเว็บนี้ ต้องเป็นเวอร์ชัน 24.13 ขึ้นไป เพราะโปรเจกต์ใช้ฐานข้อมูลที่มา กับ Node รุ่นใหม่

ขั้นที่ 2

ติดตั้ง Ollama

macOS

1. เข้าเว็บ **ollama.com/download** กด Download for macOS
2. เปิดไฟล์ที่ได้ ลาก Ollama ไปไว้ในโฟลเดอร์ Applications
3. เปิดแอป Ollama ถ้าถามให้ติดตั้ง command line ให้กด Install
4. จะเห็นไอคอนรูปลาม่าที่แถบเมนูมุมขวาบน

Windows

1. เข้าเว็บ **ollama.com/download** กด Download for Windows
2. ดับเบิลคลิกไฟล์ **OllamaSetup.exe** กด Install
3. รอติดตั้งเสร็จ Ollama จะเปิดเอง
4. จะเห็นไอคอนรูปลาม่าที่มุมขวาล่างข้างนาฬิกา

ตรวจสอบว่าใช้ได้ (เหมือนกันทั้งสองระบบ)

เปิดเบราว์เซอร์ แล้วเข้า `http://127.0.0.1:11434`

ต้องขึ้นข้อความ **Ollama is running**

ห้ามเปิดตัวเลือก "Expose Ollama to the network" ในหน้าตั้งค่าของ Ollama และห้ามตั้ง `OLLAMA_HOST=0.0.0.0` ให้ Ollama ใช้ได้เฉพาะในเครื่องนี้เท่านั้น

อธิบาย

Ollama คือโปรแกรมที่รัน AI ในเครื่องเราเอง ข้อความแชทจึงไม่ถูกส่งออกไปที่ cloud ต้องเปิด Ollama ทิ้งไว้ทุกครั้งที่ใช้เว็บ

ขั้นที่ 3

วางโปรเจกต์และเปิดหน้าต่างคำสั่ง

ฝั่งผู้ส่ง (เครื่องเรา): แฝกไฟล์โดยไม่เอาข้อมูลในเครื่องไปด้วย

```
cd /path/to/Sala
zip -r hotel-concierge.zip . -x "node_modules/*" "data/*" "backups/*"
".env" ".env.test" "*.DS_Store"
```

ห้ามส่ง `data/` `backups/` `.env` `.env.test` เพราะมีฐานข้อมูลและบัญชีพนักงานของเครื่องเรา

macOS

- ดับเบิลคลิก **hotel-concierge.zip** จะได้โฟลเดอร์ `hotel-concierge` ย้ายไปไว้ใน Documents
- เปิด Terminal พิมพ์ `cd` แล้วเว้นวรรคหนึ่งครั้ง
- ลากโฟลเดอร์ `hotel-concierge` มาวางในหน้าต่าง Terminal แล้วกด Enter

Windows

- คลิกขวา **hotel-concierge.zip** → Extract All → Extract
- เปิดโฟลเดอร์ที่แตกแล้วให้เห็นไฟล์ `package.json`
- คลิกที่ช่องที่อยู่ (address bar) ด้านบน พิมพ์ `cmd` แล้วกด Enter

ติดตั้ง library (เหมือนกันทั้งสองระบบ)

```
npm ci
```

รอจนเสร็จและกลับมาที่บรรทัดพิมพ์คำสั่ง

อธิบาย

ทุกคำสั่งหลังจากนี้ต้องพิมพ์ในหน้าต่างคำสั่งที่อยู่ในโฟลเดอร์โปรเจกต์ ถ้าปิดหน้าต่างไป ให้เปิดใหม่ด้วยวิธีเดียวกันนี้

`npm ci` ติดตั้ง library ตามเวอร์ชันที่ล็อกไว้ ทุกเครื่องจึงได้เวอร์ชันเดียวกัน

ขั้นที่ 4

ดาวน์โหลดโมเดลภาษา

เครื่อง	โมเดลที่แนะนำ	ขนาดไฟล์
RAM 16 GB (Mac หรือ Windows)	qwen3:8b (เริ่มต้นแนะนำ)	ประมาณ 5 GB
RAM 32 GB ขึ้นไป	qwen3:30b-a3b (แปลดีกว่า)	ประมาณ 19 GB
Windows ที่ไม่มีการ์ดจอ NVIDIA และตอบช้ามาก	qwen3:4b (เร็วกว่า แต่แปลแม่นน้อยกว่า)	ประมาณ 3 GB

พิมพ์ในหน้าต่างคำสั่ง (เหมือนกันทั้งสองระบบ)

```
ollama pull qwen3:8b
ollama list
```

จดชื่อโมเดลตามที่ `ollama list` แสดง (เช่น `qwen3:8b`) ต้องใช้ในขั้นที่ 5 แบบตรงทุกตัวอักษร

ผลทดสอบในรายงานของเราใช้โมเดลอีกตัว (Qwen3.6-35B-A3B ผ่าน llama.cpp) โมเดลที่เลือกจะได้ผลต่างออกไป **ต้องรันชุดทดสอบในขั้นที่ 7 ใหม่เสมอ** และยังเป็นโมเดลทดสอบ ไม่ใช่โมเดลที่อนุมัติใช้กับโรงแรมจริง

อธิบาย

โมเดลยิ่งใหญ่อยิ่งแปลแม่น แต่ใช้ RAM มากและตอบช้ากว่า เริ่มจากตัวที่เครื่องรับไหวก่อน ถ้า RAM ไม่พอ เครื่องจะช้ามากหรือโมเดลโหลดไม่ขึ้น

ขั้นที่ 5

ตั้งค่าไฟล์ .env.test

macOS

```
cp .env.test.example .env.test
open -e .env.test
```

ไฟล์จะเปิดใน TextEdit

Windows

```
copy .env.test.example
.env.test
notepad .env.test
```

ไฟล์จะเปิดใน Notepad

แท็บ 3 บรรทัดนี้ แล้วบันทึก (Mac: `⌘ + S` · Windows: `Ctrl + S`)

```
LLM_PROVIDER=ollama
LLM_BASE_URL=http://127.0.0.1:11434
LLM_MODEL=qwen3:8b
```

บรรทัดอื่นคงไว้ตามเดิม โดยเฉพาะ `OPERATING_MODE=sandbox` `LLM_TESTER_ONLY=true`
`AI_TIMEOUT_MS=30000`

บรรทัด	ความหมาย
LLM_PROVIDER	ใช้ Ollama (ในไฟล์ตัวอย่างเป็น llamacpp ของเครื่องเรา)
LLM_BASE_URL	ที่อยู่ของ Ollama ในเครื่อง
LLM_MODEL	ชื่อโมเดลจากขั้นที่ 4 ต้องตรงกับ <code>ollama list</code>
LLM_TESTER_ONLY=true	เลือกให้ใช้ได้เฉพาะโหมด sandbox กับ AI ในเครื่องเท่านั้น

อธิบาย

ไฟล์ที่ขึ้นต้นด้วยจุดจะถูกซ่อนในหน้าต่างไฟล์ จึงเปิดผ่านคำสั่งแทน การเปลี่ยน AI ไม่ต้องแก้โค้ด
แก้แค่ไฟล์นี้

ขั้นที่ 6

สร้างบัญชีผู้ดูแลและเปิดเว็บ

พิมพ์ในหน้าต่างคำสั่ง (เหมือนกันทั้งสองระบบ)

```
npm run setup  
npm run start:tester
```

`npm run setup` ทำครั้งเดียว ระบบจะถามชื่อผู้ใช้และรหัสผ่าน 14 ตัวอักษรขึ้นไป (ตอนพิมพ์รหัสจะไม่แสดงบนจอ)

`npm run start:tester` เปิดเว็บ ต้องเปิดหน้าต่างนี้ทิ้งไว้ ถ้าจะหยุดเว็บให้กด `Ctrl` + `C`

ใช้งานในเบราว์เซอร์

1. เข้า `http://localhost:3000/staff` แล้วล็อกอินด้วยบัญชีที่เพิ่งสร้าง
2. ตั้งชื่อโรงแรมที่การ์ด "บริการและนโยบาย" ช่อง "ชื่อโรงแรม"
3. ไปที่ "การเข้าพัก" เลือกห้อง แล้วเช็กอิน จดรหัสที่แสดงขึ้นครั้งเดียว
4. เปิดลิงก์ QR ของห้องในหน้าต่างส่วนตัว แล้วกรอกรหัส เพื่อทดลองเป็นแขก

macOS

หน้าต่างส่วนตัว: `⌘` + `Shift` + `N`
(Chrome / Safari)

Windows

หน้าต่างส่วนตัว: `Ctrl` + `Shift` + `N`
(Chrome / Edge)

ใช้ `http://localhost:3000` เท่านั้น ถ้าเปิดด้วย `127.0.0.1:3000` ระบบจะไม่ยอมให้ส่งข้อความ (error `origin_rejected`)

ขั้นที่ 7

ทดสอบว่าระบบทำงานจริง

ถ้าเว็บเปิดค้างอยู่ ให้เปิดหน้าต่างคำสั่งใหม่อีกหน้าต่างในโฟลเดอร์โปรเจกต์ (วิธีเดียวกับขั้นที่ 3) คำสั่งเหมือนกันทั้ง Mac และ Windows

1) ทดสอบโค้ด (ไม่ต้องใช้ AI)

```
npm test
```

ต้องผ่าน 56 จาก 56

2) ทดสอบครบวงจรกับ AI จริง

```
node --env-file=.env.test scripts/smoke-tester.js
```

ผลลัพธ์ต้องมีคำว่า `"status": "passed"` (ใช้ฐานข้อมูลชั่วคราว ไม่แตะข้อมูลในเว็บ)

3) ชุดทดสอบ 12 ภาษา

```
npm run evaluate:multilingual -- docs/multilingual-results-mine.json
```

ใช้เวลาประมาณ 5-15 นาที ส่งไฟล์ `docs/multilingual-results-mine.json` กลับมาให้ตรวจ

อธิบาย

ข้อ 1 ตรวจสอบว่าโค้ดไม่พัง ข้อ 2 ตรวจสอบว่าเว็บคุยกับ Ollama ได้จริง ข้อ 3 วัดว่าโมเดลที่เลือกแปลดีแค่ไหน

ขั้นที่ 8

แก้ปัญหาที่พบบ่อย

อาการ	วิธีแก้
ขึ้นว่าไม่รู้จักคำสั่ง <code>node</code> <code>npm</code> หรือ <code>ollama</code>	ปิดหน้าต่างคำสั่งแล้วเปิดใหม่ ถ้ายังไม่หายให้รีสตาร์ทเครื่อง บน Mac ต้องเปิดแอป Ollama อย่างน้อยหนึ่งครั้ง
<code>Windows</code> ขึ้นว่า running scripts is disabled	กำลังใช้ PowerShell อยู่ ให้เปิด Command Prompt แทน โดยพิมพ์ <code>cmd</code> ในช่องที่อยู่ของโฟลเดอร์ (ขั้นที่ 3)
ข้อความขึ้นว่า AI ไม่พร้อม / แอปไม่ได้	เปิด <code>http://127.0.0.1:11434</code> ดูว่าขึ้น "Ollama is running" ตรวจสอบชื่อใน <code>LLM_MODEL</code> ให้ตรงกับ <code>ollama list</code> แล้วกด "แอปอีกครั้ง"
ครั้งแรกช้ามาก หรือ timeout	โมเดลกำลังโหลด ลองใหม่อีกครั้ง ถ้ายังช้า เพิ่ม <code>AI_TIMEOUT_MS</code> ได้สูงสุด 60000 หรือเปลี่ยนเป็นโมเดลเล็กลง
<code>Temporary tester requires a loopback LLM endpoint</code>	<code>LLM_BASE_URL</code> ต้องเป็น 127.0.0.1 หรือ localhost เท่านั้น
<code>origin_rejected</code>	เปิดเว็บด้วย <code>http://localhost:3000</code> เท่านั้น
พอร์ต 3000 ถูกใช้อยู่ (<code>EADDRINUSE</code>)	ปิดโปรแกรมที่ใช้พอร์ตนั้น หรือแก้ทั้ง <code>PORT</code> และ <code>PUBLIC_ORIGIN</code> ใน <code>.env.test</code> เป็นพอร์ตเดียวกัน เช่น 3001
error เกี่ยวกับ <code>node:sqlite</code>	Node.js เก่าเกินไป กลับไปทำขั้นที่ 1

เมื่อขั้น 1-7 ผ่านแล้ว ให้นำ prompt 6 ส่วนในหน้าแรก ๆ ไปใช้กับ AI coding agent บนเครื่องนี้ เพื่อเปลี่ยนชื่อโรงแรมได้เลย