ภาพรวมโครงการ Hourz (Hyperlocal Service Marketplace)

แก้ไขล่าสุด 28/9/2568

1. แนวคิดหลัก (Core Concept)

ชื่อโครงการ: Hourz (อ้างอิงถึงการแลกเปลี่ยนบริการรายชั่วโมง)

วัตถุประสงค์: สร้างแอปพลิเคชันตลาดบริการในชุมชนท้องถิ่น (Hyperlocal Service Marketplace) ที่เน้นความยืดหยุ่นและความเป็นมิตร โดยใช้เทคโนโลยี Geospatial Matching (PostGIS) เพื่อเชื่อมต่อผู้คนในละแวกใกล้เคียงให้ช่วยเหลือซึ่งกันและกันได้

Core Identity:

1. **Dual Role (บทบาทคู่):** ผู้ใช้ทุกคนสามารถเป็นได้ทั้ง **"ผู้จ้าง" (Seeker)** ที่โพสต์คำขอความช่วยเหลือ และ **"ผู้ช่วย" (Helper)** ที่รับงานบริการในบัญชีเดียว
2. **Hyperlocal Matching:** การค้นหาทั้งหมดต้องอ้างอิงจาก **Fixed Location (ตำแหน่งหลัก)** ที่ผู้ใช้ปักหมุดไว้เท่านั้น

2. โครงสร้างฟีเจอร์หลัก (Detailed Feature Breakdown)

2.1 การลงทะเบียนและการยืนยันตัวตน (Onboarding & Verification)

ฟีเจอร์	รายละเอียด	วัตถุประสงค์
Authentication	ใช้ระบบ JWT ของ FastAPI (Mock Google Sign-in) ในการลงทะเบียน/เข้าสู่ระบบ	รักษาความปลอดภัยของบัญชี
ยืนยันตำแหน่งหลัก	**(บังคับ)** ผู้ใช้ต้องปักหมุด **Fixed Location** ที่อยู่ของตนเองลงบนแผนที่ (บันทึกพิกัดด้วย **PostGIS**)	ใช้เป็นจุดศูนย์กลางในการคำนวณระยะทางและค้นหา
ยืนยันตัวตน	**(เสมือน)** อัปโหลดเอกสารยืนยันตัวตน (ID Verification Mock)	เพิ่มความน่าเชื่อถือให้กับโปรไฟล์ผู้ใช้
Availability Toggle	**สวิตช์ "พร้อมรับงาน"**	ควบคุมบทบาท Helper

	บนหน้าโปรไฟล์ ผู้ใช้สามารถเปิด/ปิดสถานะ **is_available** เพื่อรับ Notification ของ Gig Requests ได้	ได้ตลอดเวลา

2.2 การลงรายการบริการ (Service Listing: Gigs & Requests)

ประเภทรายการ	**ผู้โพสต์**	**รายละเอียดที่สำคัญ**
Gig Request	Seeker	คำขอจ้างงาน, รายละเอียด, ราคา/งบประมาณ, **ตำแหน่งงานที่ปักหมุด** และ **จำนวนผู้ช่วยที่ต้องการ (N)**
Helper Gig	Helper	รายละเอียดบริการที่ตนเองเสนอ (เช่น เดินสุนัข), **Hourly Rate/Fixed Price**
ระบบค้นหา	ใช้การกรองตาม **คำหลัก, หมวดหมู่, Rate, และระยะทาง (PostGIS)**	เน้นการค้นหางาน/บริการที่อยู่ใกล้ที่สุด

3. กลไกการจับคู่และรับงาน (Booking Pool Flow)

กระบวนการนี้จะเกิดขึ้นเมื่อ **Seeker** โพสต์งาน **Request** และ **Helper** สนใจรับงานนั้น

ขั้นตอน	**ผู้ดำเนินการ**	**รายละเอียดการทำงาน**
1. โพสต์ Request	Seeker	กำหนดจำนวนผู้ช่วยที่ต้องการ (N)
2. จองคิว (Book Gig)	Helper	กดปุ่ม **"จองคิวรับงาน"** เพื่อเข้าสู่ **Pool** ของ Request นั้น ๆ
3. Pool System	ระบบ	เก็บรายชื่อ Helper

		ที่จองคิวทั้งหมด (ไม่จำกัดจำนวน) แต่จะแสดงแจ้งเตือน Seeker หาก Pool มี Helper เพียงพอแล้ว
4. เจรจาและคัดเลือก	Seeker	ทบทวนโปรไฟล์/Reputation Score ของ Helper ใน Pool จากนั้น **ทักแชท** เพื่อตกลงรายละเอียดและราคา
5. Final Acceptance	Seeker	**กดปุ่ม "ยืนยันรับงาน"** ในห้องแชท เพื่อเลือก Helper คนนั้น
6. Update Status	ระบบ	สถานะงานเปลี่ยนเป็น **Accepted** และจำนวนผู้ช่วยที่ต้องการของ Request นั้นจะลดลง (N-1)
7. Notification	ระบบ	แจ้งเตือน Helper คนอื่นที่ยังอยู่ใน Pool ว่า **"งานนี้มีคนรับไปแล้ว"**

4. ปรัชญาการออกแบบ UI/UX (Dual Role Solution)

เราใช้ **Mixed UI** เพื่อให้ Hourz ใช้งานง่ายและมีความยืดหยุ่นสูง โดยใช้ **Toggle Switch** และ **Tab Bar** เป็นเครื่องมือหลักในการจัดการ Content

4.1 Bottom Navigation Bar (4 Tabs หลัก) (คร่าวๆ)

แถบนำทางหลักต้องมี 4 เมนู: **[Home]**, **[Chat]**, **[Orders]**, **[Profile]**

4.2 การสลับบทบาทในหน้า Home

- **Header Toggle:** มี **Toggle Switch** ที่ชัดเจนอยู่ด้านบนของหน้าจอ
  - **โหมด Seeker:** แสดง Content ที่เป็น **Helper Gigs** (บริการที่ฉันสนใจจะจ้าง)
  - **โหมด Helper:** แสดง Content ที่เป็น **Gig Requests** (งานที่ฉันสนใจจะรับ)
- **Action Button:** ปุ่ม **Floating Action Button (FAB) [+ โพสต์]**
  เมื่อกดแล้วต้องมีเมนูให้เลือกว่าจะโพสต์ **Request** หรือ **Gig**

4.3 ระบบจัดการบัญชีและกิจกรรม

- **หน้า Profile:** ใช้ **Tab Bar** ภายในเพื่อแยกประวัติกิจกรรมออกจากกันอย่างชัดเจน: **[งานที่ฉันจ้าง]** และ **[งานที่ฉันรับ]**
- **Availability Toggle:** ปุ่ม **is_available** สำหรับ Helper ต้องแสดงผลชัดเจนบนหน้า Profile เพื่อให้สามารถสลับสถานะรับงานได้อย่างรวดเร็ว

4.4 ระบบความปลอดภัย (Safety & Reporting)

- **ระบบรายงาน:** ต้องมีปุ่ม **"รายงาน"** ให้เข้าถึงได้ง่ายบนหน้า **โปรไฟล์ผู้ใช้คนอื่น** และใน **ห้องแชท** เพื่อให้ผู้ใช้สามารถรายงานพฤติกรรมที่ไม่เหมาะสมได้อย่างรวดเร็ว

5. Technical Stack

องค์ประกอบ	**เทคโนโลยีที่ใช้**	**บทบาทใน Hourz**
Backend API	FastAPI (Python)	API หลักในการจัดการ Logic ทั้งหมด
Database	PostgreSQL with **PostGIS**	จัดการข้อมูล, ระบบ Queue, และ Queries เชิงพื้นที่
Chat System	FastAPI **WebSockets**	การสื่อสารแบบ Real-time (ใช้สำหรับการเจรจาต่อรอง)
Frontend	Flutter/Dart	พัฒนาแอปพลิเคชันบน Mobile (iOS/Android)
Authentication	JWT (JSON Web Tokens)	การยืนยันตัวตนและการจัดการ Session
Transaction	**Mock Payment System**	ระบบจำลองการจ่ายเงิน/หักค่าธรรมเนียมบริการ