Prompt 1b: Saved Listings / Wishlist Feature

"Add a feature that allows guests to save or bookmark property listings they are interested in. Saved listings should be easily accessible from the guests' dashboard under a dedicated ‘Saved Listings’ or ‘Wishlist’ section.

- Users can add or remove listings from their saved list with a single click.

- Display saved listings with key details: thumbnail, title, location, price, and status.

- Support sorting and filtering saved listings by date saved, price, or proximity.

- Persist saved listings across sessions and devices.

- Ensure the UI is intuitive and responsive on both desktop and mobile.

Prompt 1c: Dashboards for Landlord & Tenant

Create detailed, engaging dashboards for RooMe’s two user roles with a focus on usability and personalization:

Host Dashboard:

- List all their property listings with status (active, pending, expired)

- Ability to add, edit, delete listings, including posting individual rooms with dynamic labeling (e.g., kitchen, dining, bedroom) tailored to property types (house, boarding house, motel, etc.)

- View detailed stats per listing (number of views, inquiries, saves)

- Show verification status and badge prominently

- Notifications overview (new inquiries, messages, verification updates)

- Settings area for profile, payment info, and notification preferences

- Intuitive navigation menus and quick action buttons (e.g., “Add Listing”)

- Mobile-first responsive design with clean, modern UI and subtle animations to enhance engagement

Analytics & Insights

1. Track Metrics:

- Views per listing: Count each time a user views a listing.

- Saves: Count the number of users who save/bookmark a listing.

- Messages Sent: Track how many times a landlord was contacted through the app.

2. Data Storage:

- Store these metrics in the backend

3. Landlord Dashboard UI:

- Display weekly or monthly stats:

Example:

- "Your listing ‘Villa Heights’ had 53 views this week."

- "12 users saved this listing."

- "3 inquiries received."

General UI Aesthetic

- Mobile-first, modern, minimal

- Rounded elements, clean spacing

- Soft animations (e.g., stat count-up, smooth tab transitions)

- Use RooMe brand colors and typography

- Dark/light mode toggle

Bottom Navigation Bar (5 Main Tabs)

| Icon | Tab | Function |

|------|------|----------|

| 🏠 | Dashboard | Overview, quick stats, alerts |

| 🏘️ | Listings | Manage property listings |

| 📊 | Insights | Views, saves, inquiries |

| 💬 | Messages | User communication |

| ⚙️ | More | Profile, settings, payments |

🏠 \*Dashboard Tab (Main Screen)

- Welcome Banner: “Hi, Ngoni 👋”

- Verification Badge if verified

- Quick Stats Cards (horizontally scrollable):

- Active Listings

- New Messages

- Total Views this Month

- Pending Verifications

- Notifications List:

- "New inquiry on 'Villa Heights'"

- "Your listing is now verified!"

- Quick Actions:

- ➕ Add New Listing

- 📈 View Insights

- ✏️ Edit Profile

Listings Tab

- Segmented Controls:

- All | Active | Pending | Expired

- Each Listing Card Shows:

- Thumbnail image + name

- Status badge (Active, Pending, Expired)

- Type tag (House, Boarding House, Motel)

- View & edit buttons

- Add New Listing Button (Floating FAB)

Inside Add/Edit Listing:

- Dynamic room labeling

- Photo uploads per room

- Amenities toggle

- Description, price, availability dates

Insights Tab

- Time filter: Weekly | Monthly

- For each listing:

- Views: "53 this week"

- Saves: "12 users saved"

- Inquiries: "3 contacts"

- Line chart or bar graph display

- Tap a listing to view detailed metrics

Messages Tab

- Inbox layout (list of conversations)

- New/unread badge indicator

- Tap to open full chat

- Predefined quick replies (optional)

More Tab (Settings/Profile)

- Profile Info

- Add/Edit Payment Method

- Notification Preferences

- Contact Support

- Logout

Guest Dashboard:

- Display personalized greeting with name and profile picture

Key Sections and Features to Include:

Top Search Bar

- Prominent search input with placeholder: “Search for spaces, services, or experiences…”

-filter icon.

Search Functionality

UI:

- Top bar with search field and optional filters (location, category, price).

Technical Needs:

- Full-text search via Supabase/PostgreSQL.

- Location filtering using GPS (user permission needed).

- Search index for performance (use Supabase extensions like `pg\_trgm` or integrate Algolia for advanced search).

2. Main Navigation Tabs (Sticky or Scrollable)

- Tabs: Homes | Marketplace | find a Roommate

- Each tab loads tailored content (e.g. Homes = rooms, lodges; Marketplace = handymen, decor; find a roommate = roommate matching and searching).

UI:

- Scrollable tabs or segment controls that update the listing feed below.

Technical Needs:

- Dynamic queries by category from the backend.

- Category tags stored in DB for each listing.

- Tabs filter the `listings` table by `type` column.

Perfect, now that we’re using \*this Airbnb-like aesthetic\* with \*main tabs at the top (Homes, Marketplace, Roommates)\* and preserving the \*bottom navigation\*, here’s the refined \*UI layout and functionality for each top tab\*, while keeping your design language consistent:

---

\*1. Homes Tab (Default/Selected)\*

\*Goal:\* Help users search, discover, and book housing.

\*Top Area:\*

- \*Search bar\* (rounded, full-width): “Start your search” with location suggestions.

- \*Main Tabs:\*

- Homes (active)

- Marketplace

- Roommates

\*Body:\*

- \*Continue Search Section\*

- Shows last searched location with date preview (e.g., Harare)

- \*Recently Viewed Homes\*

- Horizontal scroll of property cards

- Info: image, location, beds, rating, save (heart icon)

- \*Recommended / Nearby Sections\*

- “Stay near [Location]”

- Grid/list of cards with images, tags like “Guest Favourite”

---

\*2. Marketplace Tab\*

\*Goal:\* Showcase furniture, appliances, and home services.

\*Top Area:\*

- Same search bar

- Main tabs: Homes | \*Marketplace\* (active) | Roommates

Body:

- \*Continue Browsing Section\* (if any previous category viewed)

- \*Category Quick Filters:\*

- Furniture | Solar & Energy | Appliances | Services

- Each with icons or chips

[22/08, 11:57] Sora: - \*Recently Viewed / Popular Products\*

- Cards with:

- Product Image

- Title & price

- Seller rating

- Location

- Save icon

- Promoted Items Section

3. Roommates Tab

\*Goal:\* Match users with compatible roommates or shared spaces.

Top Area:

- Same search bar

- Main tabs: Homes | Marketplace | \*Roommates\* (active)

Body:

a. Consent + Toggle Area

- Message: “Show my profile to potential roommate matches.”

- \*Toggle ON/OFF\*

- When ON:

- Enable viewing others

- Enable notifications for recommendations

- Enable match suggestions

- When OFF:

- Disable all above

b. Current Roommate Matches Section

- If already matched:

- Show profile pic + name

- Shared space details

- “Message” or “View Agreement” buttons

c. Suggested Roommates

- Tinder-style or vertical cards:

- Name, age, location, budget

- Tags (Smoker, Night Owl, Clean, etc.)

- Match % score

- Action: “Connect” / “Pass”

d. Filters Section

- Gender, Budget, Lifestyle, Preferred Areas

- Mini floating button to refilter

e. Notifications Section

- “You’ve got a new match!”

- “5 people matched your roommate preferences.”

Bottom Navigation (Untouched):

- Explore | Wishlists | Trips | Messages | Profile

Perfect — here’s how we’ll \*make the top search bar dynamic per tab\* while keeping the layout consistent:

Dynamic Top Search Bar Behavior (Across Tabs)

Design:

- \*Appearance:\* Same rounded rectangle with a search icon.

- \*Position:\* Fixed at the top, under the logo/header.

- \*Behavior:\* The \*placeholder text changes dynamically\* based on the active tab.

Tab 1: Homes

- \*Placeholder:\* `Search for places to stay...`

- \*Search Scope:\* Cities, towns, property types (e.g., “House in Waterfalls”)

Tab 2: Marketplace

- \*Placeholder:\* `Search furniture, solar, cleaning & more...`

- \*Search Scope:\* Services, product categories, vendors (e.g., “Solar installer” or “Couches”)

Tab 3: Roommates

- \*Placeholder:\* `Search roommates by area, age, lifestyle...`

- \*Search Scope:\* Profiles of roommate-seeking users (filtered by toggled consent), shared spaces, preferences

Technical Note:

- Use context-aware logic to detect the active tab and update the placeholder dynamically.

- Keep the search bar component reusable but control text via state (React or Flutter logic depending on stack).

3. Dynamic Banners

- “Continue Searching in [Location]” with dates if previously selected.

- Image-based CTA banners with localised imagery (e.g. “Explore Harare’s Top Guest Rooms”).

UI:

- Dynamic banner based on previous user activity.

Technical Needs:

- Store \*last search parameters\* in a `user\_preferences` or `search\_history` table.

- Use session/localStorage or Supabase `auth.user.id` to fetch previous activity.

4. Recently Viewed Section

- Carousel of recently viewed listings (with image, location, price, star rating).

- Actionable (tap to reopen listing).

UI:

- Horizontally scrollable cards.

Technical Needs:

- Track viewed listings with timestamps.

- Use a `recent\_views` table with `user\_id`, `listing\_id`, `timestamp`.

- Fetch last 5–10 entries for current user.

5. Local Discovery Section

- Label: “Stay near [Current Suburb or Geo-based Suggestion]”

- Showcase listings nearby with “Guest Favourite” tags or hearted items.

- Listings display: image, name, rating, 1-liner about the host or service.

UI:

- Location-based listings sorted by proximity.

Technical Needs:

- Ask for GPS access from user.

- Store `lat/lon` for all listings in the database.

- Use \*geospatial queries\* (Haversine formula or Supabase PostGIS extension) to find nearby rooms/services.

6. Suggested for You / Popular Picks

- Based on past interaction or general trends.

- Can include rooms, services, or experiences mixed.

UI:

- Personalized feed (ML-lite).

Technical Needs:

- Store interaction data: views, likes, bookings.

- Build a basic recommendation engine

- Popular = sorted by likes/bookings.

- Personalized = based on past category views.

- Store in Supabase or mirror to Redis for fast access.

7. Bottom Navigation Bar

- Tabs: Explore | Wishlist | Bookings | Messages | Profile

- Icons styled minimal, easily tappable, highlight active tab.

UI:

- 5-tab structure: Explore | Wishlist | Bookings | Messages | Profile.

Technical Needs:

- Tab routing handled via app framework (Flutter, React Native, etc.).

- Each tab is a component/screen with its own backend calls.

Wishlist (Heart Icon)

UI:

- Heart icon on each card for saving.

Technical Needs:

- `wishlist` table: `user\_id`, `listing\_id`.

- Display wishlist items in a separate screen.

Messaging System

UI:

- Simple inbox UI with recent conversations.

Technical Needs:

- Realtime messaging: use \*Supabase Realtime\* or third-party (e.g. Firebase or Stream).

- Store chats in `messages` table with `sender\_id`, `receiver\_id`, `timestamp`, `text`.

Booking System

UI:

- Tap into listing → book now button → select dates → confirm.

Technical Needs:

- `bookings` table: `user\_id`, `listing\_id`, `start\_date`, `end\_date`, `status`.

- Date pickers (Flutter or Web component).

- Logic to prevent double bookings (conflict check).

Profile Picture and Personalization

UI:

- Avatar display for hosts/users.

Technical Needs:

- File uploads to Supabase Storage or Cloudinary.

- Link image URL to `profiles` table.

Notifications

UI:

- Bell icon or in-app popups.

Technical Needs:

- Use Supabase Functions or cron jobs for scheduled notifications (e.g. booking reminders).

- Store notification history in a `notifications` table.

8. Color and Typography

- Use modern, warm tones suited for the African/Zim context.

- Clean font (e.g. Inter, SF Pro, or Roboto), large enough for comfort.

9. Other UX Notes:

- Smooth scrollable sections.

- Clear visual hierarchy.

- Quick actions (e.g. heart to wishlist, message host).

- Display location context (e.g. city/suburb) clearly on cards

Prompt 1d: Dynamic Listing Forms for Landlords

Create dynamic listing forms for landlords to add or edit properties on RooMe. The form must adapt based on the selected property type (boarding house, motel, hotel, apartment, lodge) and include a universal “Highlights & Extras” multi-select tag system for all property types.

Form Requirements:

- Mandatory fields (all types):

- Title

- Location

- Price (numeric validation)

- Description

- Property type selection

- Optional fields (all types):

- Amenities (checkboxes)

- Images (with preview before submission)

- Specific mandatory fields by property type:

- Boarding house / Motel / Hotel:

- Room count

- Bed types

- Check-in and check-out times

- Lodge:

- Outdoor facilities (e.g., pool, BBQ area)

- Available activities (e.g., hiking, fishing)

- Form Behavior:

- Multi-step with progress indicator

- Dynamic validation for all inputs (required, numeric, etc.)

Support saving drafts and auto-saving periodically to prevent data loss

- Dynamic section toggles (e.g., show room details only when relevant)

- Duplicate listing detection warning if title and location match existing listings

- Preview mode before final submission

- Auto-generate map pin from location input (using Google Maps API or similar)

Universal Highlights & Extras Tag Selector:

- Add a multi-select tag input for “Highlights & Extras” that applies to all property types.

- Tags display as clickable, toggleable hashtag-style buttons (similar to Facebook post hashtags).

- Multiple selections allowed with clear visual feedback.

- Tags appear on listing cards and detail pages as small badge-style elements to communicate key amenities/perks.

- Include a search/filter input above the tag list for easy navigation.

- The tag list should be easily extendable as RooMe grows.

Examples of initial curated tags:

#SwimmingPool, #FreeWiFi, #Gym, #PetFriendly, #SecureParking, #LaundryService, #CloseToTransport, #Garden, #Balcony, #24hrSecurity, #StudyArea

Prompt 2: Detailed Messaging System for RooMe (iMessage Style)

Message Initiation:

- guests can send inquiries directly from any property listing page via a clearly visible ‘Message Landlord’ button.

- Landlors receive new message alerts linked specifically to the listing the inquiry concerns.

Message Storage & Structure:

- Messages are tied to both the specific listing and the user pair (tenant-landlord).

- Store all messages securely in the database with timestamps, sender ID, receiver ID, and linked listing ID.

- Support text messages with optional emoji support.

Message Threads & History:

- Organize messages into threaded conversations per listing between tenant and landlord.

- Users can view complete message history per conversation within the last 6 months; older messages archived but searchable.

UI & UX Features (iMessage Vibe):

- Use distinct rounded chat bubbles with subtle shadows — blue for sent messages, gray for received — mimicking iMessage style.

- Smooth animations for sending, receiving, and loading messages, including typing indicators.

- Message status indicators: sent, delivered, read — subtle and intuitive.

- Show user display names in message headers: guests use their usernames; landlords show business or registered names.

- Include smooth scrolling, quick reply gestures, and responsive input fields.

Messaging Icon Placement:

- Prominently display messaging icon on each listing page near landlord info.

- Also place in user dashboards for easy access to all conversations, with notification badges showing unread counts.

Notifications:

- Push in-app notifications and optional email alerts for new messages.

Security & Privacy:

- Only authorized users can access their conversations.

- Sanitize inputs to prevent security risks.

Mobile & Accessibility:

- Fully responsive UI optimized for mobile and desktop.

- Accessible with screen readers and keyboard navigation.

Prompt 3

Add a feature allowing guests (Guests) to leave a rating (1–5 stars) and a written review for a listing after an interaction. Requirements:

- Reviews are tied to the specific listing and tenant user.

- Rating (1–5 stars) is mandatory; review text is optional but encouraged, with a max length limit.

- Reviews go live immediately

- Landlords cannot remove or edit reviews; once posted, reviews stay permanently.

- guests can edit or delete their own reviews only within 7 days of posting.

- Display all reviews on the listing page showing tenant username, rating stars, date, and review text.

- Reviews directly impact the listing’s overall trust score.

- Support sorting reviews by date, rating, or relevance.

- Show average rating prominently on listing cards and detail pages along with the total reviews count for context.

- Notify landlords when new reviews are posted.

- Ensure the review system is responsive and accessible on mobile and desktop.

- Implement a star rating input UI with hover and touch effects for improved usability across devices.

- Allow guests to mark reviews as “helpful” with a thumbs-up icon to enable better sorting and highlight useful feedback.

Prompt 4: Pulsar and Smart AI search

Develop a smart search engine for RooMe called Pulsar, powered by GPT-4 (here is the API key:), enabling users to search using natural language queries like:

“Affordable room near Avondale with Wi-Fi and private bathroom.”

Core Functionality:

- Use GPT-4 or open-source embedding models to convert both user queries and listing data into vector embeddings.

- Store listing vectors using pgvector in PostgreSQL.

- Pulsar should match user queries to listings based on semantic relevance, not just keywords.

Data used for semantic embeddings:

- Listing title

- Description

- Amenities and highlights/extras (tags)

- Location details

- Landlord profile bio or business name

- Rating and Reviews

Search Output:

- Rank results by semantic similarity, then refine with:

- Proximity (distance from user’s query location or GPS)

- Trust score (boost verified or high-rated listings)

- Price relevance (if mentioned in the query or filter)

Filters Supported:

- Property type (e.g., boarding house, motel, apartment)

- Price range

- Rental duration

- Verification status (optional toggle)

Behavior:

- If \*no matches\*, display: “No results found. Try adjusting filters or keywords.

- If \*weak matches\*, optionally show nearby or similar results labeled as:

- “Suggested Alternatives”

- “Worth a Look”

Examples: Pulsar should offer alternatives like: no room matching your needs in avondale, however, there's a two roomed house costing 200, all bills inclusive in (maybe an area in the same location or vicinity with similarities to the desired area). This is to prevent blunt responses but offer human like suggestions. Another example is: No such house in Mabvuku, but there's one in Tafara with even better pricing. Mabvuku and Tafara are of almost similar proximity to town.

(Only if relevance threshold is met — never force irrelevant results.)

Performance & Logging:

- Must return results in \*under 500ms\* for smooth UX.

- Log queries and outcomes for analytics and future improvement.

- Cache frequently searched queries for faster response.

Additional Notes:

- All listings and bios should be pre-embedded and stored.

- Pulsar should automatically refresh embeddings when listings or bios are updated.

- Design result UI to clearly differentiate exact vs suggested matches.

Smart Search & Pulsar UI Details

- Central, rounded search bar with a subtle Pulsar icon inside (left or right).

- Dynamic placeholder text cycling through example Pulsar queries (e.g., “Single room near CBD with Wi-Fi”).

- Real-time Pulsar suggestions dropdown as user types.

- Next to the search bar, a small toggle button labeled “Manual Search” or an icon (e.g., filter sliders) to switch search modes.

- When toggled ON, manual search panel appears beside or below the search bar with:

- Keyword input with clear/reset buttons

- Advanced filters: property type, price range, rental duration, amenities, etc.

- Message: “Use manual search if you want precise control or Pulsar can’t find what you need.”

- When toggled OFF, manual search panel hides, Pulsar suggestions resume.

- Search results area updates based on active mode, with a clear visual indicator showing whether Pulsar or Manual search is active.

- Pulsar results include subtle “Suggested Alternative” labels for relevant but non-exact matches.

- UI elements use rounded corners, clean spacing, and subtle animations to match RooMe’s modern aesthetic.

Manual search

Implement a basic filter-based search system for RooMe’s listing interface. Users should be able to manually search for listings by applying specific filters like \*location\*, \*price range\*, \*listing type\* (e.g., lodge, apartment, boarding), and \*amenities\*. Use simple dropdowns, checkboxes, or input fields in the frontend (HTML/CSS), and process the filter inputs using Flask on the backend. The backend should query a predefined dataset (can be hardcoded or stored in a JSON/DB file), filter based on the selected criteria, and return matching listings to the user.

Display the results below the filter section, showing key details like title, location, and price. This is the foundation of the search function and should be expandable later with dynamic data and enhanced UI.

Prompt 5: Location-Aware Smart Search with Pulsar

- When user grants location access:

- Detect location via browser/device geolocation API.

- Prioritize nearby listings in search results by combining semantic relevance and proximity ranking.

- If location access denied/unavailable:

1. Prompt user to manually enter a preferred location.

2. Offer recent search locations saved locally or linked to user account.

3. Default to Harare CBD if no input.

UI Details:

- Show message near search bar: “Showing listings near you” when location active.

- Provide a visible “Change Location” button to allow manual override anytime.

- Use reverse geocoding to display readable location names, not just lat/lng.

- On location update, refresh search results dynamically.

Integration:

- Seamlessly integrate location logic with GPT-4-powered semantic search (Prompt 4).

- Rank listings by combined semantic relevance, trust score, and geographic proximity.

- Ensure smooth UX on desktop and mobile with clear location controls.

Prompt 7: Trust Score System & Animated Badges

A. Responsiveness to Inquiries (Total: +15 Points)

Scoring Breakdown:

1. Average Response Time (within 1–3 hours) — +7\*

- Encourages landlords to reply quickly, improving user experience.

2. Response Rate (percentage of inquiries responded to) — +5\*

- High response rate shows engagement and reliability.

3. First Response Promptness (responding within the first few hours of initial contact) — +3\*

- Critical for converting leads and maintaining interest.

What Replit Should Build:

- Background job to track timestamps of inquiry and reply

- Metric calculation engine to update responsiveness scores

- Frontend indicators like “Typically responds within X hours”

B. Document Verification (Total: +20 Points)

Ensures the legitimacy of listings and builds trust between users and landlords.

Scoring Breakdown:

1. National ID Upload (Landlord identity) — +8

- Confirms landlord is a reerson or business.

2. Property Ownership Proof (e.g. title deed, lease) — +8

- Verifies that the landlord has legal rights to list the property.

3. Additional Supporting Docs (e.g. utility bill, company registration) — +4

- Strengthens authenticity and trust score.

What Replit Should Build:

- Secure document upload system (file encryption & verification flags)

- Admin panel to manually review and approve/reject documents

- Auto-score updates on successful verifications

- Visual badge/icon for verified listings

C. Engagement Score Breakdown (Total: +10 Points)

Scoring Logic:

1. Listing Views (per 100 views) — (+4 Points max)

- Scaled by traffic:

- 0–50 views: +0

- 51–100 views: +2

- 100+ views: +4

2. Saves (number of users saving to wishlist) — (+4 Points max)

- 1–4 saves: +2

- 5+ saves: +4

3. Click-throughs to Contact Landlord — (+2 Points)

- Tracks how many users hit “Contact” or similar CTA.

What Replit Should Build:

- Backend counters for:

- Views per listing

- Saves per listing

- Contact clicks

- Auto-score recalculation (daily or on action)

- Flags for top-performing listings

- Optional: decay older engagement over time for relevance

D. RooMe Profile Completeness Scoring (Total: +10 Points)

Scoring Breakdown:

1. Profile Picture (Logo or Headshot) — (+2 Points)

- Uploading a clear image or logo.

2. Cover Photo — (+1 Point)

- Optional but adds visual credibility.

3. Business Name / Display Name — (+1 Point)

- Properly filled and not just default email.

4. Bio or Description — (+2 Points)

- Minimum word count (e.g., 20–50 words) introducing the landlord or property agent.

5. Contact Information Verified — (+2 Points)

- Verified phone number and/or email.

6. Location Set — (+1 Point)

- City, suburb, or base location properly filled.

7. Social Links or External Website (if any) — (+1 Point)

- Optional but useful for trust and brand extension.

What Replit Should Build:

- Form with progress tracking (e.g., "Profile 70% complete")

- Backend fields & validation

- Track what's missing for scoring.

- Score calculation function

- Triggered on profile update or dashboard view.

E. RooMe Listing Freshness Scoring (Total: +15 Points)

Scoring Breakdown:

1. Newly Posted Listings — (+7 Points)

- Listings posted within the last 14 days get full points.

- Score decays over time (e.g., 1 point reduction per week after day 14).

2. Recently Updated Listings — (+5 Points)

- Listings with updates to price, availability, images, or description within \*last 10 days\*.

- Encourages landlords to keep details fresh.

3. Active Availability Sync — (+3 Points)

- Listings that consistently reflect current availability (e.g., not marked as booked when still visible).

- Can be tied to landlord login activity or periodic system prompts.

What Replit Should Build:

1. Database fields for timestamps:

- `created\_at`, `last\_updated\_at`, `last\_availability\_check`

2. Cron job or scheduler:

- To \*evaluate age and updates\* of listings on a rolling basis.

3. Score decay function:

- Time-based algorithm to reduce freshness score as days pass.

F. Rating and reviews +28 points

1. Verified Reviewer Check — (+10 Points)

- Ensure only \*registered and confirmed users\* can submit reviews.

- Implement backend logic to \*validate user status\* before accepting review input.

2. Review Volume Tracker — (+5 Points)\*

- Track number of approved reviews per listing.

- Listings with more reviews gain \*greater scoring weight\*.

3. Recency Checker — (+5 Points)

- Check \*review date.

- Only reviews from the last 6 months contribute full points.

4. Rating Consistency Evaluator — (+5 Points)

- Measure rating stability (e.g., minimal fluctuation over 10+ reviews).

- Reward consistent average ratings (e.g., 4.2–4.6 stars).

5. Textual Review Analyzer — (+3 Points)

- Check for descriptive content in the review.

- Minimum character count + presence of keywords like clean, quiet, close to town, etc.

Badge Levels:

- Platinum (80–100)

- Gold (70–79)

- Silver (60–69)

- None (<60)

Badge Design & Placement:

- Show badges next to the landlord’s profile image or business logo on listing cards — small, elegant, and non-intrusive.

- On dashboard and profile pages, display badges larger.

- Clicking or tapping a badge reveals its name and explanation via a smooth slide or fade animation.

Visual Style:

- Platinum: Electric blue shimmer with animated glow

- Gold: Gradient gold with soft pulsing effect

- Silver: Frosted silver with a slight shimmer

- All icons should use sleek, minimal visuals — no cartoon-like designs.

- Hover/tap animations reinforce quality and credibility.

User Experience Goals:

- Make badges feel rewarding and aspirational.

- Avoid UI clutter — badges must support trust-building without overwhelming the interface.

- System should be easy to update or expand (e.g. future badge types like “Top Responder” or “Highly Rated”).

Prompt 8: Document Verification with OCR + AI

"Implement a document verification system for RooMe:

- \*Upload Requirements:\* Landlords can upload up to three documents for verification:

- Government-issued ID

- Recent utility bill

- Title deed or lease agreement

OCR Integration:

- Use OCR (mock or actual API like Google Vision or Tesseract) to extract text from uploaded documents.

- Parse extracted data for key identifiers: name, address, ID number, etc.

AI Verification Logic:

- Send parsed OCR text to GPT-4 or other language model.

- Compare against the user’s RooMe profile details (name, contact info, etc).

- Determine match confidence.

UI & Feedback:

- Show verification status (Pending, Verified, Rejected) in the Landlord dashboard.

- Notify user upon verification result.

Badge Integration:

- Once verified, award a \*Verified\* badge (linked to the trust score system).

- Display badge on profile and listing cards.

Ensure system is secure, easy to use, and flexible enough to switch OCR/LLM providers later."

Prompt 9: In-App Notifications System

Add an in-app notifications system to RooMe for both Landlords and Guests. Include the following event triggers:

- New message received

- Listing approved or rejected

- Listing nearing expiry (e.g., 3 days before)

- New inquiry received

- New review posted

- Verification status update

- Trust score change or badge upgrade

Features:

- Display notifications in a bell icon dropdown (top-right or header bar)

- Show unread indicator and allow marking as read

- Group notifications by date

- Store in database and allow access to notification history

- Add setting toggles in user profile for enabling/disabling types of notifications

- Support optional email and/or push notifications (scaffold with placeholders for future integration)

- For mobile PWA/native later: use service workers for push support

Ensure the system is responsive, lightweight, and extensible.

Prompt 11 – Overlay Text Animation on Listing Cards

"Add an animated text overlay to each property listing card on RooMe. The overlay should appear on hover (desktop) or subtly after 1 second of view (mobile), then shrink or slide up after 2–3 seconds. It should display concise info in this format:

`[Location] • [Boarding house, lodge, motel, hotel, residence]`

Examples:

- “Avondale, Harare • Hotel”

- “CBD, Harare • Boarding House”

- “Greendale, Harare • New”

If no badge is assigned, fallback to “New” or show only location.

Style it with a soft black translucent background, white text, rounded edges, and smooth transitions. Keep it subtle, modern, and mobile-friendly."

RooMe User Interaction Data Capture Plan (with guest/landlord separation)

1. Events to Track:

- Listing views (which listing, when, by whom)

- Listing saves/bookmarks

- Messages sent (to landlords or support)

- Reviews submitted (rating, text, timestamp)

- Log-in and log-out timestamps (session durations)

- Smart search queries (query text, timestamp, results clicked)

- Manual/filter-based searches (filters used, timestamp, results clicked)

- Clicks on ads or affiliate links

- Feature usage (which features used, duration of use)

- User navigation paths (pages/screens visited, time spent)

2. Database Schema Suggestions:

Users Table:

- user\_id

- user\_type (enum: ‘guest’, ‘landlord’)

- profile info + session meta

Listings Table:

- listing\_id

- property details + metadata

UserInteractions Table:

- interaction\_id (PK)

- user\_id (FK)

- user\_type (redundant but helpful for fast filtering)

- event\_type (view, save, message, review, search\_smart, search\_manual, ad\_click, feature\_use, etc.)

- event\_data (JSON: listing\_id, query\_text, filters, etc.)

- timestamp

- Log events asynchronously to avoid UI delays

- Use timestamps, session IDs, and user\_type to segment analytics by guest vs landlord

- Aggregate and analyze data daily/weekly/monthly for trends and insights

- Index on user\_id, user\_type, event\_type, timestamp for efficient querying

- Store event\_data flexibly in JSON to accommodate varied event details

FOR THE FUTURE

Prompt 1: Ads & Affiliate Integration with Slot-Based Sales

Integrate a flexible advertising system in RooMe that supports native ads and affiliate marketing with the following features:

1. Ad placements:

- Between property listings in browsing/search feeds

- Within blog content and static pages

- In user dashboards (both Landlord and Guest) as non-intrusive sidebar or banner ads

- On Explore or Recommendations pages as featured/promoted content

- Dedicated ‘Featured Services’ section for businesses buying premium slots

2. Ad sales model:

- Allow external advertisers (e.g., service providers like solar companies, movers) to purchase fixed ad slots on specific pages or sections

- Slots can be time-bound or impression-based

- Admin panel to manage ad inventory, sales, scheduling, and performance tracking

3. Ad types:

- Native ads styled consistently with RooMe’s UI

- Banner ads, sponsored listings, and affiliate links/banners

- Optionally, rewarded ads users can watch for perks

4. Fallback and empty states:

- If no ads are sold for a slot, show a clean placeholder or promote RooMe’s own services/promotions

- Ensure user experience remains smooth with no dead spaces or intrusive ads

5. Analytics:

- Track impressions, clicks, conversions per ad

- Provide advertisers with performance reports via admin dashboard

6. Affiliate marketing:

- Seamlessly embed referral links and banners within blog posts and static pages

- Track affiliate link clicks and conversions

Ensure all ads and affiliate content maintain RooMe’s design consistency, responsiveness, and performance."

Prompt 2: Remote Admin API

Build a secure REST API endpoint `/admin/dashboard` that returns comprehensive data for RooMe’s remote admin panel. The response should include:

1. User & Interaction Analytics:

- Total number of users, split by type: guests, landlords

- Active users (last 7 days) by user type

- User session stats: average session duration, number of sessions

- Interaction counts over selectable date ranges, separated by user type:

- Listing views

- Listing saves/bookmarks

- Messages sent

- Reviews submitted

- Smart searches performed

- Manual/filter searches performed

- Ad clicks

- Feature usage

2. Listings & Verification:

- Total listings count, new listings (last 7 days)

- Listings pending verification and their statuses

- Verified landlords and their document verification statuses

- Flagged or rejected documents with reasons

3. Admin Actions & Moderation:

- Recent manual approvals/rejections of listings and documents

- Content flags and their resolution statuses

- User bans, suspensions, and warnings issued

4. Ads & Affiliate Management:

- Current ad inventory with details (location, size, duration, status)

- Active and scheduled ads with performance metrics (impressions, clicks)

- Affiliate link and banner management data (performance stats)

- Controls for creating, updating, pausing, or removing ads and affiliate content

5. Additional Requirements:

- Support pagination and filtering (by date, user type, status)

- Return JSON data grouped by logical sections (`users`, `interactions`, `listings`, `documents`, `adminActions`, `ads`, `affiliate`)

- Ensure data security and efficient query performance

Prompt 3 - Interactive Maps Feature — RooMe User Workflow & Implementation Prompt

User Workflow:

1. Search & Browse:

- User enters a search query or uses filters (location, price, type) on RooMe’s main search page or using smart search

2. View on Map Option:

- Below or alongside the listing results, a \*“View on Map”\* button is shown.

- When clicked, RooMe loads an interactive map view centered on the search location or user location (if allowed)

3. Map Interaction:

- Listings from the search appear as pins on the map.

- User can pan, zoom, and click pins to open popups showing brief details and a link to full listing.

- Map filters (price, type, distance radius) update visible pins dynamically.

4. Opt-out or Continue Browsing:

- User can close the map and return to the card/tile view anytime.

- Alternatively, clicking a card from the original list opens the full listing details page.

Technical Implementation Details:

1. Front-end:

- Search page shows results as cards with “View on Map” button.

- Map view component uses Google Maps or Mapbox JS SDK to render map and pins.

- Filter controls on map update backend query to fetch filtered listings dynamically.

2. Back-end:

- API endpoint accepts search params + filters, returns listings with geo-coordinates.

- Geocoding service converts location keywords into lat/lng for map centering.

- Implement session or state management to remember user choices (map view or list view).

3. APIs to integrate:

- Google Maps JS API or Mapbox GL JS: Map rendering, pins, popups.

- Google Maps Geocoding API or Mapbox Geocoding API: Convert location searches.

- \*Browser Geolocation API:\* For optional user location detection and map centering.

- RooMe Backend API: To fetch listings data based on filters and location.

Sample Flow of API Calls:

- User searches "lodges near Masvingo Polytechnic"

- Frontend calls RooMe backend: `GET /api/listings?location=Masvingo+Polytechnic&filters=...`

- Backend calls Geocoding API: converts "Masvingo Polytechnic" → lat,lng

- Backend queries database for listings within radius of lat,lng with filters

- Backend returns listings with lat,lng to frontend

- User clicks “View on Map” → map centers on lat,lng, pins plotted for each listing

- User adjusts map filters → frontend calls backend again with updated filters