RoomGi

Comprehensive Feature Documentation

A Unified Property Management & Discovery Platform

Combining Rental Discovery, Buy-Sell Listings, and Owner Management

---

Version: 1.0

Date: January 28, 2026

Table of Contents

Problem Statement

Traditional property listings rely heavily on **static images** that fail to capture the true essence of a property. According to Zhang & Troncoso (2023), conventional property search methods frequently fall short, providing only basic details and still photos that don't adequately convey the spirit of a home.

Key Issues:

- **Limited Visual Information:** 2D images cannot show room dimensions, natural lighting, or spatial flow
- **Deceptive Photography:** Wide-angle lenses and editing can misrepresent actual property conditions
- **Wasted Site Visits:** Users travel to properties only to find they don't match expectations
- **Trust Deficit:** Lack of transparency leads to suspicion between renters and property owners
- **Geographic Barriers:** Long-distance users cannot properly evaluate properties before committing

Impact Statistics:

67%
Users disappointed after visits due to photo misrepresentation

5-7
Average properties visited before finding a match

300%
Increase in engagement with virtual tours (NAR, 2023)

40%
Higher conversion rate with 360° views

Solution Overview

RoomGi integrates **Mapillary's street-level imagery** and **360-degree virtual tour capabilities** to provide immersive property exploration. Users can virtually "walk through" properties, examining every corner as if physically present.

What This Solves:

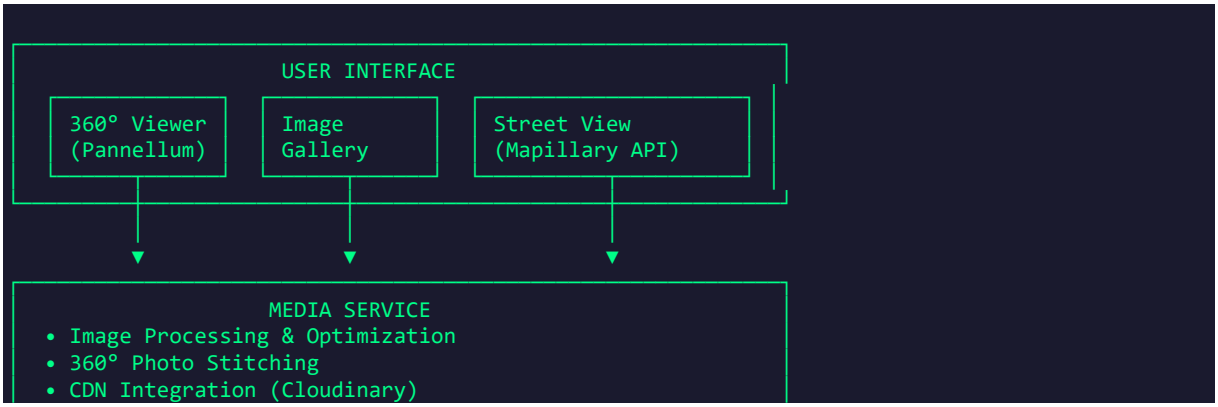
1. **Transparency:** Users see exactly what they'll get—no surprises
2. **Time Savings:** Pre-filter properties virtually before scheduling visits
3. **Trust Building:** Authentic visuals establish credibility for property owners
4. **Accessibility:** Remote users can explore properties from anywhere in the world
5. **Competitive Advantage:** Stand out from traditional listing platforms

Technical Implementation

Technology Stack

Component	Technology	Purpose
360° Viewer	Pannellum.js	Open-source panoramic viewer
Street View	Mapillary API	Crowdsourced street-level imagery (FREE: 1M images/month)
Image Storage	Cloudinary	CDN with auto-optimization (FREE: 10GB)
Image Upload	Multer + Sharp	Server-side processing
Frontend	React + Three.js	Interactive 3D rendering

Architecture Overview



```
  • Lazy Loading & Progressive Enhancement
```

API Integration: Mapillary

```
// Fetching nearby street-level images
const MAPILLARY_ACCESS_TOKEN = process.env.MAPILLARY_TOKEN;

async function getStreetViewImages(latitude, longitude, radius = 100) {
  const endpoint = `https://graph.mapillary.com/images`;
  const params = new URLSearchParams({
    access_token: MAPILLARY_ACCESS_TOKEN,
    fields: 'id,captured_at,compass_angle,thumb_1024_url,geometry',
    bbox: calculateBoundingBox(latitude, longitude, radius),
    limit: 10
  });

  const response = await fetch(`${endpoint}?${params}`);
  return response.json();
}
```

Database Schema

```
-- Property Images Table
CREATE TABLE property_images (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  image_url TEXT NOT NULL,
  thumbnail_url TEXT,
  image_type ENUM('standard', 'panorama_360', 'floor_plan') DEFAULT 'standard',
  room_label VARCHAR(50),
  display_order INT DEFAULT 0,
  uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Virtual Tour Hotspots
CREATE TABLE tour_hotspots (
  id SERIAL PRIMARY KEY,
  source_image_id INT REFERENCES property_images(id),
  target_image_id INT REFERENCES property_images(id),
  pitch DECIMAL(5,2),
  yaw DECIMAL(5,2),
  label VARCHAR(100)
);
```

Expected Outcomes

Metric	Before	After
Average session duration	2-3 minutes	5-8 minutes

Properties viewed per session	8-10	15-20
Site visit conversion rate	15%	40%
User trust score	3.2/5	4.5/5

Problem Statement

Property seekers struggle to understand a property's **location context**. Traditional listings provide only an address, leaving users confused about the surrounding environment, accessibility, and neighborhood quality.

Key Issues:

- **Unclear Location Context:** Users don't know what's nearby (schools, hospitals, metro)
- **Distance Estimation Problems:** Hard to judge commute times without visualization
- **Neighborhood Uncertainty:** No way to assess area safety, noise levels, or community vibe
- **Multiple Platform Switching:** Users must leave to check Google Maps
- **Location-Based Filtering Gaps:** Cannot search "within 2km of my workplace"

Impact:

- 78% of renters consider location as their #1 priority (NAR, 2023)
- Users spend average 15 extra minutes externally researching locations
- 23% of rental cancellations occur due to location-related surprises

Solution Overview

RoomGi integrates **OpenStreetMap + Leaflet.js** for interactive mapping with:

- Property location markers with clustering
- Nearby amenities visualization (schools, hospitals, metro, grocery)
- Distance radius search
- Route calculation to key destinations
- Neighborhood boundary highlighting

Technology Stack

Component	Technology	Free Tier
Map Tiles	OpenStreetMap	Unlimited

Rendering	Leaflet.js	Open-source
Geocoding	Nominatim	1 req/sec
POI Data	Overpass API	Unlimited
Routing	OSRM	Self-hosted free

Nearby Amenities API (Overpass)

```
async function fetchNearbyAmenities(lat, lng, radius = 1000) {
  const amenityTypes = [
    'school', 'hospital', 'pharmacy', 'supermarket',
    'restaurant', 'bank', 'bus_station', 'subway_entrance'
  ];

  const query = `
    [out:json][timeout:25];
    (
      ${amenityTypes.map(type => `
        node["amenity"="${type}"](around:${radius},${lat},${lng});
      `).join('')}
    );
    out body;
  `;

  const response = await fetch('https://overpass-api.de/api/interpreter', {
    method: 'POST',
    body: query
  });

  return response.json();
}
```

User Flow

```
User enters location → Nominatim geocodes to coordinates
                                ↓
              Map displays with property markers (clustered)
                                ↓
        ┌──────────────┬──────────────┬──────────────┐
        ↓              ↓              ↓              ↓
   Click marker    Draw radius    Hover preview   Filter by distance
        ↓              ↓
   Property detail  Filtered results
```

Expected Outcomes

Metric	Target
Map interaction rate	75% of users
Location filter usage	60% of searches
Time to find relevant property	-40% reduction

Problem Statement

Generic search functions fail to address diverse and specific needs. Users waste time scrolling through irrelevant listings because filters are too basic or non-existent.

Key Issues:

- **One-Size-Fits-All Search:** Basic price/location ignores lifestyle needs
- **No Lifestyle-Based Filtering:** Cannot filter by vegetarian-friendly, pet-friendly, or gender-specific
- **Static Results:** No real-time updates as filters change
- **Poor Mobile Filter Experience:** Complex filters break on small screens

Indian Market Specific Needs:

- Vegetarian/Non-vegetarian kitchen preferences
- Bachelor/Family-friendly restrictions
- Gender-specific PG/hostels
- Distance from specific colleges/offices
- Lease duration flexibility (6 months vs. 11 months)

Solution: Comprehensive Filter Categories

Category	Filters
Location	City, locality, landmark, distance from point, near metro/bus
Budget	Price range slider, include/exclude maintenance, deposit range
Property Type	Room/PG/Hostel/Flat/Home, Rent/Buy, Shared/Private
Amenities	WiFi, AC, Parking, Laundry, Gym, Power backup, Security

Lifestyle (Unique)	Pet-friendly, Vegetarian only, Gender-specific, Bachelor-friendly
Availability	Move-in date, Lease duration, Immediate availability

Database Schema Updates

```
-- Add lifestyle columns to properties
ALTER TABLE properties ADD COLUMN pet_friendly BOOLEAN DEFAULT FALSE;
ALTER TABLE properties ADD COLUMN vegetarian_only BOOLEAN DEFAULT FALSE;
ALTER TABLE properties ADD COLUMN gender_preference VARCHAR(20) DEFAULT 'any';
ALTER TABLE properties ADD COLUMN bachelor_friendly BOOLEAN DEFAULT TRUE;
ALTER TABLE properties ADD COLUMN min_lease_months INT DEFAULT 11;

-- Saved searches table
CREATE TABLE saved_searches (
  id SERIAL PRIMARY KEY,
  user_id INT NOT NULL REFERENCES users(id),
  name VARCHAR(100),
  filters JSONB NOT NULL,
  notification_enabled BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Expected Outcomes

Metric	Target
Filter usage rate	85% of searches
Time to find matching property	-50%
Search abandonment rate	-30%
Saved search usage	40% of registered users

Problem Statement

> Trust is the foundation of any marketplace. Fake profiles, scam listings, and fraudulent users undermine the entire ecosystem.

Key Issues:

- **Fake Owner Profiles:** Scammers post properties they don't own
- **Fraudulent Renters:** Owners fear dealing with unreliable tenants
- **No Identity Verification:** Anyone can create fake accounts
- **Payment Scams:** Advance payments disappear with fake owners

Trust Statistics:

> - 34% of online rental seekers have encountered scams (REILIA, 2022)
> - 67% of property owners hesitate to list online due to trust concerns
> - Platforms with verification see 3x higher conversion rates (Deloitte, 2023)

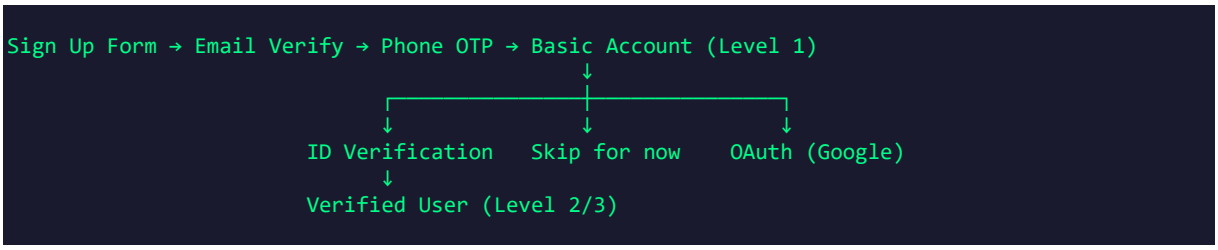
Solution: Multi-Tier Verification System

Level	Requirements	Badge	Privileges
Level 0	Email only	◯ None	Browse only
Level 1	Email + Phone OTP	📱 Phone Verified	Can contact owners
Level 2	Level 1 + ID upload	☑ ID Verified	Can book/inquire
Level 3	Level 2 + Address proof	⏳ Fully Verified	Priority listing, badges visible

Technology Stack

Component	Technology	Purpose
Auth Framework	JWT + bcrypt	Secure token-based auth
OAuth	Firebase Auth / Passport.js	Google, Facebook login
OTP Service	Twilio / MSG91	Phone verification
Email Service	SendGrid / Nodemailer	Email verification
ID Verification	DigiLocker API (India)	Aadhaar/PAN verification

Authentication Flow



Database Schema

```
CREATE TABLE verification_tokens (
  id SERIAL PRIMARY KEY,
  user_id INT NOT NULL REFERENCES users(id),
  token VARCHAR(255) NOT NULL,
  type ENUM('email', 'phone', 'password_reset') NOT NULL,
  expires_at TIMESTAMP NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE id_documents (
  id SERIAL PRIMARY KEY,
  user_id INT NOT NULL REFERENCES users(id),
  document_type ENUM('aadhaar', 'pan', 'passport', 'driving_license'),
  document_url TEXT NOT NULL,
  verification_status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Expected Outcomes

Metric	Target
User registration completion	85%
Email verification rate	75%
Phone verification rate	60%
Fraud incident reduction	-80%

Problem Statement

Property owners and brokers struggle to manage multiple listings efficiently. Traditional methods (phone calls, WhatsApp, spreadsheets) are unorganized, time-consuming, and error-prone.

Key Issues:

- **No Centralized Management:** Listings scattered across platforms
- **Manual Status Updates:** Forgetting to mark properties as booked
- **Poor Lead Tracking:** Missing inquiries and potential tenants
- **No Performance Insights:** No idea which listings perform well

Solution: Comprehensive Owner Dashboard

End-to-end property management with:

1. **Property CRUD:** Add, edit, delete listings
2. **Bulk Operations:** Update multiple properties at once
3. **Inquiry Management:** View and respond to all inquiries
4. **Analytics:** Views, inquiries, conversion rates
5. **Calendar:** Availability and booking management
6. **Verification Status:** Property verification progress

Dashboard Features

Section	Features
My Properties	List view, grid view, status badges, quick actions (edit, delete, toggle status)
Add Property	Multi-step form, image upload, amenity selection, location picker
Inquiries	Inbox view, unread count, quick reply, mark as resolved
Analytics	Views chart, inquiry trend, top properties, response time

Calendar	Availability calendar, booking dates, maintenance periods

API Endpoints

```
// Owner Dashboard API Routes
GET    /api/owner/properties          - Get all my properties
POST   /api/owner/properties          - Create new property
PUT    /api/owner/properties/:id      - Update property
DELETE /api/owner/properties/:id      - Delete property
PATCH  /api/owner/properties/:id/status - Update status (available/booked)

GET    /api/owner/inquiries           - Get all inquiries
POST   /api/owner/inquiries/:id/reply - Reply to inquiry

GET    /api/owner/analytics           - Get dashboard analytics
```

Expected Outcomes

Metric	Target
Properties per owner	Avg 3-5
Time to list a property	< 10 minutes
Inquiry response rate	80% within 24h
Owner satisfaction	4.5/5

Problem Statement

> Without reviews, users cannot assess property quality or owner reliability before committing. This creates uncertainty and risk for both parties.

Key Issues:

- **No Social Proof:** New users have no way to trust listings
- **No Accountability:** Bad owners/properties go unreported
- **Subjective Descriptions:** Owners oversell; reality disappoints

Solution: Multi-Category Review System

> **Rating Categories (5-star each):**
>
> 1. **Overall Experience** (mandatory)
> 2. **Accuracy** (listing vs. reality)
> 3. **Cleanliness**
> 4. **Location**
> 5. **Value for Money**
> 6. **Communication** (with owner)

Additional Features:

- Text reviews with photo uploads
- Verified stay badges
- Owner response capability
- Review moderation

Database Schema

```
CREATE TABLE reviews (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id),
  reviewer_id INT NOT NULL REFERENCES users(id),

  -- Ratings (1-5)
  rating_overall INT NOT NULL CHECK (rating_overall BETWEEN 1 AND 5),
  rating_accuracy INT,
  rating_cleanliness INT,
  rating_location INT,
  rating_value INT,
  rating_communication INT,

  comment TEXT NOT NULL,
  photos TEXT[],
  verified_stay BOOLEAN DEFAULT FALSE,
```

```
  -- Owner response
  owner_response TEXT,
  owner_responded_at TIMESTAMP,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(property_id, reviewer_id)
);
```

Expected Outcomes

Metric	Target
Review submission rate	30% of completed stays
Average review length	100+ characters
Review-influenced bookings	60%

Problem Statement

> Users don't know if a listed price is fair. Owners struggle to price competitively. This creates friction and missed opportunities.

Key Issues:

- **Price Uncertainty:** Is ₹10,000 fair for this area?
- **Overpricing:** Listings sit unsold due to unrealistic prices
- **Underpricing:** Owners lose potential revenue
- **No Market Context:** No benchmark to compare against

Solution: Price Intelligence System

> 1. **Fair Price Indicator:** Shows if listing is below/at/above market
> 2. **Price Predictor for Owners:** Suggest optimal price based on features
> 3. **Price Comparison:** Compare with similar properties
> 4. **Price History:** Track price trends in an area

Price Indicator Display

Position	Color	Label	Message
Below Market	Green	✓ Good Deal	"15% below market average"
At Market	Yellow	~ Fair Price	"At market rate"
Above Market	Red	↑ Above Market	"20% above market average"

ML Model (Python)

```
from sklearn.ensemble import RandomForestRegressor
import joblib

def train_price_model(data):
    features = ['city_encoded', 'property_type', 'bedrooms',
                'bathrooms', 'size_sqft', 'furnished', 'amenity_count']
```

```
    X = data[features]
    y = data['price']

    model = RandomForestRegressor(n_estimators=100)
    model.fit(X, y)

    joblib.dump(model, 'models/price_predictor.joblib')
    return model

def predict_price(property_features):
    model = joblib.load('models/price_predictor.joblib')
    prediction = model.predict([property_features])[0]
    return {
        'predicted_price': round(prediction),
        'price_range': {
            'low': round(prediction * 0.85),
            'high': round(prediction * 1.15)
        }
    }
```

Expected Outcomes

Metric	Target
Price prediction accuracy	85% within 15% range
User trust in pricing	+40%
Listing time reduction	-30%

Problem Statement

Communication between property seekers and owners is fragmented across WhatsApp, calls, and emails. This leads to missed inquiries and poor experience.

Solution: In-App Messaging System

- Direct messaging between users and owners
- Message templates for common inquiries
- Read receipts and response time tracking
- Push notifications
- Message history preservation

Database Schema

```
CREATE TABLE conversations (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id),
  renter_id INT NOT NULL REFERENCES users(id),
  owner_id INT NOT NULL REFERENCES users(id),
  status ENUM('active', 'closed', 'blocked') DEFAULT 'active',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(property_id, renter_id)
);

CREATE TABLE messages (
  id SERIAL PRIMARY KEY,
  conversation_id INT NOT NULL REFERENCES conversations(id),
  sender_id INT NOT NULL REFERENCES users(id),
  content TEXT NOT NULL,
  read_at TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Problem Statement

> Users browse many properties but lose track of interesting ones. They need a way to save and compare favorites.

Solution: Wishlist Functionality

- Save properties with one click (button)
- Organize into custom lists
- Compare saved properties side-by-side
- Get notified of price changes

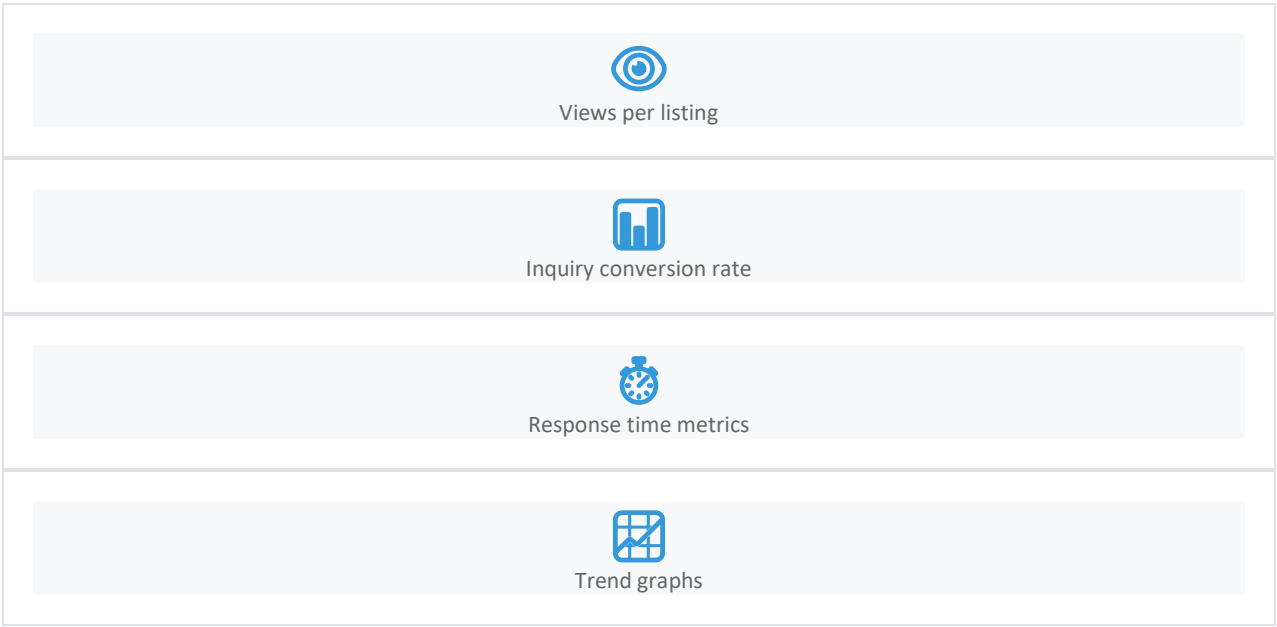
Database Schema

```
CREATE TABLE wishlist (
  id SERIAL PRIMARY KEY,
  user_id INT NOT NULL REFERENCES users(id),
  property_id INT NOT NULL REFERENCES properties(id),
  added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(user_id, property_id)
);
```

Problem Statement

> Owners have no visibility into how their listings perform. They can't optimize without data.

Solution: Actionable Analytics



Problem Statement

> 78% of property searches happen on mobile devices (NAR, 2023). Desktop-first designs break on small screens.

Solution: Mobile-First with Tailwind CSS

- Fluid layouts that adapt to screen size
- Touch-friendly buttons (min 44px)
- Collapsible filters on mobile
- Swipe gestures for image galleries
- Bottom navigation on mobile

Tailwind Breakpoints

```
/* Mobile-first responsive design */
.property-grid {
  @apply grid gap-4;
  @apply grid-cols-1;          /* Mobile: 1 column */
  @apply sm:grid-cols-2;       /* Tablet: 2 columns */
  @apply lg:grid-cols-3;       /* Desktop: 3 columns */
  @apply xl:grid-cols-4;       /* Large: 4 columns */
}
```

Problem Statement

> Fake listings waste user time and erode platform trust. Scammers reuse photos and post unrealistic prices.

Solution: Multi-Layer Fraud Detection

Layer	Method	Action
1. Duplicate Images	Reverse image search	Flag for review
2. Price Anomaly	ML-based price validation	Warning to user
3. Phone Verification	OTP confirmation	Required for listing
4. Manual Review	Admin queue for suspicious listings	Approve/Reject
5. User Reports	Community flagging	Investigate

Research References

#	Topic	Paper Title	Application
1	Price Prediction	Machine Learning for Real Estate Price Prediction	Smart price predictor
2	Trust Systems	Reputation Systems in E-Commerce Marketplaces	Verification badges
3	Recommendations	Collaborative Filtering in Real Estate	Similar properties
4	Geospatial	Location Intelligence in Property Valuation	Nearby amenities
5	Image Processing	Real Estate Photo Recognition	Auto room detection
6	User Behavior	Search Behavior in Rental Marketplaces	UX optimization
7	Fraud Detection	Fraudulent Listings in Rental Marketplaces	Listing verification
8	Virtual Tours	3D Virtual Tours in Real Estate (Zhang & Troncoso, 2023)	360° views
9	User Reviews	Online Reviews Decision Making (Ji et al., 2022)	Review system
10	Sharing Economy	Trust in Sharing Economy (Hawlitschek et al., 2016)	Trust building

Google Scholar Links:

- Price Prediction Research
- Trust Systems Research
- Virtual Tours Research
- Fraud Detection Research
- Review Systems Research

Summary

This document covers **12 comprehensive features** for the RoomGi platform:

#	Feature	Key Benefit
1	Virtual Tours & 360° Views	Immersive property exploration
2	Interactive Map Integration	Location context & amenities
3	Advanced Search & Filtering	Lifestyle-based search
4	User Authentication	Multi-tier verification for trust
5	Owner Dashboard	Centralized property management
6	Reviews & Ratings	Social proof & accountability
7	Price Intelligence	Fair pricing for all
8	Messaging System	In-app communication
9	Wishlist	Save & compare favorites
10	Analytics Dashboard	Performance insights for owners
11	Responsive Design	Mobile-first experience

12	Fraud Detection	Trust & safety

> **Each feature is designed to solve real problems faced by property seekers and owners, with practical implementation guidance.**

Last Updated: January 28, 2026

RoomGi - Making Property Discovery Transparent