🏠 RoomGi - Unified Property Management & Discovery Platform

A hackathon project combining rental discovery, buy-sell listings, and owner management into one powerful platform

---

1. PROJECT OVERVIEW (5-LINE SUMMARY)

RoomGi is an all-in-one property platform that solves fragmented real estate discovery. It serves **three user groups simultaneously**:

1. Students/professionals searching for rental rooms/PGs/hostels with verified info
2. Home buyers exploring flat/home listings directly from sellers
3. Property owners/brokers managing multiple listings efficiently from one dashboard

The platform eliminates intermediaries, reduces fake listings through verification, and provides transparency with street-level views, price comparisons, and direct seller contact. It's built as a fully responsive web app with filtering, real-time updates, and a scalable architecture—not just a landing page, but a working marketplace.

2. WHAT'S BUILT vs. WHAT'S MISSING

☑ What You Need to Build:

Frontend (React + Tailwind)

- Multi-page responsive website (not SPA, actual pages)
- Property listing pages with advanced filters (location, budget, amenities, availability)
- Individual property detail pages with images gallery + street view
- Owner/Broker dashboard for CRUD operations
- User authentication (signup/login)
- Search & sort functionality
- Responsive design (mobile-first)
- Reviews & ratings system
- Contact/inquiry messaging

Backend (Node.js/Express or Python/Flask)

- RESTful API for properties (CRUD)

- User authentication & authorization
- Advanced search/filtering
- Image upload handling
- Database schema management
- Geolocation services
- Email notifications
- Analytics tracking

Database (PostgreSQL)

- Properties table (rental, buy-sell, status)
- Users table (owners, buyers, renters, brokers)
- Listings table with geo-coordinates
- Reviews/ratings table
- Images table
- Inquiries/messages table

🎯 Differentiation (Stand-Out Features):

- Street-level views via Mapillary
- Price comparison tool
- Verified owner badges
- AI-powered price prediction
- Neighborhood safety scores
- Amenity availability calendar
- Virtual tours (360° photos)
- Direct messaging with reviews

3. SIMILAR SUCCESSFUL APPS - FEATURES TO BORROW

App 1: OLX (olx.in)

Strengths to copy:

- Simple, fast search & filtering
- Trust scoring for sellers
- Category-based organization
- Mobile-friendly interface
- Posted time indicators

App 2: MagicBricks (magicbricks.com)

Strengths to copy:

- Advanced property filters (furnishing, type, size)
- Neighborhood information panels
- Price trends graphs
- Similar properties suggestions
- Posted by owner/agent distinction

App 3: 99Acres (99acres.com)

Strengths to copy:

- Property comparison tool
- Amenities checklist
- Locality ratings
- Builder/project profiles (for brokers)
- Property history tracking

App 4: Airbnb (for inspiration)

Strengths to copy:

- Photo gallery with zoom
- User reviews with photos
- Host profiles with verification badge
- Availability calendar
- Instant messaging between users

App 5: Zillow (for data viz)

Strengths to copy:

- Zestimate (price estimates)
- Price history graphs
- Market insights/trends
- Mortgage calculator
- School/neighborhood ratings

4. FREE TECH STACK & APIS

Frontend Stack

Technology	Purpose
React 18+	Component-based UI
Tailwind CSS	Styling, responsive design
Leaflet.js + OpenStreetMap	Interactive maps
React Query	Data fetching/caching
Axios	HTTP client
React Router	Multi-page navigation
Formik + Yup	Form validation
React Slick	Image carousels
Chart.js	Price trends, graphs
React Icons	Iconography

Backend Stack

Technology	Purpose

Node.js + Express.js	Lightweight & fast server
PostgreSQL	Powerful, free, reliable database
Sequelize ORM	Database queries
JWT	Authentication
Multer	File uploads
Nodemailer	Email sending
Dotenv	Environment config
CORS	Cross-origin requests

Hosting (Free Tier)

Service	Purpose
Vercel	Frontend hosting (free tier, unlimited projects)
Render.com	Backend hosting (free tier with auto-deploy)
Render PostgreSQL	Database hosting (free trial)

Cloudinary	Image Storage (free tier: 10GB storage)

APIs & Integrations (ALL FREE)

API	Purpose	Free Tier	Link
Mapillary	Street-level images inside rooms/properties	1M images/month	mapillary.com/developer
OpenStreetMap + Leaflet	Interactive mapping	Unlimited	leafletjs.com
Nominatim (OSM)	Geocoding (address → coordinates)	1 req/sec	nominatim.openstreetmap.org
Google Places API	Location suggestions & autocomplete	1000 free/month	developers.google.com
Firebase Auth	User authentication	50k logins/month free	firebase.google.com
Cloudinary	Image storage & optimization	10GB storage free	cloudinary.com

SendGrid / Mailgun	Email notifications	100 emails/day free	sendgrid.com
Chart.js	Price trends visualization	Free & open-source	chartjs.org
Unsplash API	Sample property images for demo	50 requests/hour	unsplash.com/api

5. IMPLEMENTATION ROADMAP

Phase 1: Core Setup (Week 1)

- Setup React + Tailwind frontend
- Setup Node.js + Express backend
- Setup PostgreSQL database
- Create data schema
- Setup authentication (JWT)
- Deploy skeleton to Vercel + Render

Phase 2: Basic Features (Week 1.5)

- Property listing CRUD (backend API)
- Property listing page (frontend)
- Search & filtering
- Owner dashboard basics
- Image upload

Phase 3: Advanced Features (Week 2)

- Mapillary street view integration
- Maps with property markers
- Messaging system
- Reviews & ratings
- Price comparison tool

Phase 4: Polish & Stand-Out (Week 2.5)

- Price prediction algorithm
- Neighborhood ratings
- Responsive mobile design
- Performance optimization
- Bug fixes & testing

6. RESEARCH PAPERS & GOOGLE SCHOLAR LINKS

Feature Inspiration from Academic Research:

#	Topic	Paper Title	Google Scholar Link	Feature Inspiration
1	Price Prediction Models	Machine Learning for Real Estate Price Prediction	Search Link	Implement simple price range prediction based on location/size
2	Trust & Verification Systems	Reputation Systems in E-Commerce Marketplaces	Search Link	Verified badges for owners, user reviews, rating algorithms
3	Recommendation Systems	Content-Based and Collaborative Filtering in Real Estate	Search Link	Suggest similar properties, personalized search
4	Geospatial Analysis	Location Intelligence in Property Valuation	Search Link	Show nearby amenities, distance to schools/hospitals, safety scores

5	Image Processing	Automated Real Estate Photo Recognition	Search Link	Auto-detect room features from photos (kitchen, bathroom count)
6	User Behavior Analysis	Search Behavior in Online Rental Marketplaces	Search Link	Popular searches, trending locations, seasonal patterns
7	Fraud Detection	Detecting Fraudulent Listings in Rental Marketplaces	Search Link	Flag suspicious postings, duplicate photos, pricing anomalies

7. EXTRA STAND-OUT FEATURES (TO WIN)

🚀 *Wow Features (Pick 2-3):*

1. Virtual 360° Tours

- Use Mapillary's panoramic images
- Create walkthrough experience of rooms
- Embed panoramic viewers for each property

2. Smart Price Predictor

- Analyze historical data on your platform
- Predict fair price based on: location, size, amenities, demand
- Show comparison: "₹5000 vs avg ₹4800" on listings
- Use simple ML (Linear Regression with TensorFlow.js)

3. Neighborhood Safety Score

- Aggregate user reviews mentioning safety
- Show icons: crime rate, traffic, noise level
- Link to public datasets if available

4. Availability Calendar

- PG/hostel owners set availability
- Users see real-time availability
- Color-coded: Available (green), Booked (red), Maintenance (gray)

5. Quick Contact + Messaging

- Direct in-app messaging (not email)
- Show response time ("Usually replies in 2 hours")
- Rating system for communication

6. Advanced Filters (Unique)

- Distance from metro/college
- Furnished level (unfurnished, semi, fully)
- Shared vs private
- Pet-friendly filter
- Vegetarian/non-veg community
- Gender-specific options
- Lease duration flexibility
- Amenities checklist (WiFi, laundry, parking, etc.)

7. Analytics Dashboard for Owners

- Views per listing (graph)
- Inquiries count
- Conversion rate (inquiries → bookings)
- Price comparison with nearby properties
- Seasonal demand trends

8. Verification Badge System

- Phone verification (mandatory)
- Email verification
- ID verification (if time allows)
- Show verification level on profile

9. Price Comparison Tool

- User clicks "Compare" on a property
- Show 3-5 similar properties side-by-side
- Highlight differences in amenities/price

- Why it's better/worse than alternatives

10. AI Chatbot (Optional)

- Answer common questions (check-in times, pet policy, etc.)
- Uses FAQ database
- Can be simple rule-based (no actual AI needed)

8. PROJECT STRUCTURE (File Organization)

```
RoomGi/
├── frontend/                    # React App
│   ├── public/
│   ├── src/
│   │   ├── components/          # Reusable components
│   │   │   ├── PropertyCard.jsx
│   │   │   ├── FilterPanel.jsx
│   │   │   ├── Navbar.jsx
│   │   │   └── MapViewer.jsx
│   │   ├── pages/               # Page components
│   │   │   ├── HomePage.jsx
│   │   │   ├── SearchPage.jsx
│   │   │   ├── PropertyDetail.jsx
│   │   │   ├── OwnerDashboard.jsx
│   │   │   └── LoginPage.jsx
│   │   ├── services/            # API calls
│   │   │   └── api.js
│   │   ├── App.jsx
│   │   └── index.css
│   ├── package.json
│   └── .env.local
│
├── backend/                     # Node.js + Express
│   ├── routes/
│   │   ├── properties.js        # Property CRUD
│   │   ├── users.js             # Auth & profiles
│   │   ├── listings.js          # Owner listings
│   │   └── inquiries.js         # User inquiries
│   ├── controllers/
│   │   ├── propertyController.js
│   │   ├── userController.js
│   │   └── inquiryController.js
│   ├── models/                  # Database models
│   │   ├── User.js
│   │   ├── Property.js
│   │   └── Inquiry.js
│   ├── middleware/
│   │   └── auth.js              # JWT verification
│   ├── config/
│   │   └── database.js          # PostgreSQL connection
│   ├── server.js
│   ├── package.json
│   └── .env
│
├── database/                    # SQL schemas
│   └── schema.sql
│
└── docs/                        # Documentation
```

```
    ├── API_ENDPOINTS.md
    ├── DATABASE_SCHEMA.md
    └── DEPLOYMENT.md
```

9. DATABASE SCHEMA (PostgreSQL)

```
-- Users Table
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  phone VARCHAR(15) UNIQUE,
  user_type ENUM('buyer', 'owner', 'broker', 'renter') NOT NULL,
  profile_pic_url TEXT,
  is_verified BOOLEAN DEFAULT FALSE,
  verification_level INT DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Properties Table
CREATE TABLE properties (
  id SERIAL PRIMARY KEY,
  owner_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  title VARCHAR(255) NOT NULL,
  description TEXT,
  property_type ENUM('room', 'pg', 'hostel', 'flat', 'home') NOT NULL,
  listing_type ENUM('rent', 'sale') NOT NULL,
  address VARCHAR(255) NOT NULL,
  city VARCHAR(100) NOT NULL,
  state VARCHAR(100) NOT NULL,
  postal_code VARCHAR(10),
  latitude DECIMAL(10, 8),
  longitude DECIMAL(11, 8),
  price INT NOT NULL,
  price_period ENUM('monthly', 'yearly', 'one-time') DEFAULT 'monthly',
  size INT,  -- in sq ft
  bedrooms INT,
  bathrooms INT,
  furnished ENUM('unfurnished', 'semi-furnished', 'fully-furnished'),
  available_from DATE,
  status ENUM('available', 'booked', 'maintenance') DEFAULT 'available',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Amenities Table (Join Table)
CREATE TABLE property_amenities (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  amenity VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Images Table
CREATE TABLE images (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  image_url TEXT NOT NULL,
  display_order INT,
```

```
  uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Reviews Table
CREATE TABLE reviews (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  reviewer_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  rating INT CHECK (rating >= 1 AND rating <= 5),
  comment TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Inquiries/Messages Table
CREATE TABLE inquiries (
  id SERIAL PRIMARY KEY,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  sender_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  message TEXT NOT NULL,
  status ENUM('new', 'replied', 'closed') DEFAULT 'new',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Wishlist Table
CREATE TABLE wishlist (
  id SERIAL PRIMARY KEY,
  user_id INT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  property_id INT NOT NULL REFERENCES properties(id) ON DELETE CASCADE,
  added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(user_id, property_id)
);

-- Create Indexes for Performance
CREATE INDEX idx_properties_city ON properties(city);
CREATE INDEX idx_properties_price ON properties(price);
CREATE INDEX idx_properties_owner ON properties(owner_id);
CREATE INDEX idx_images_property ON images(property_id);
CREATE INDEX idx_reviews_property ON reviews(property_id);
CREATE INDEX idx_inquiries_property ON inquiries(property_id);
```

10. KEY API ENDPOINTS

AUTH

Method	Endpoint	Description
POST	/api/auth/register	User signup
POST	/api/auth/login	User login

POST	/api/auth/logout	User logout
POST	/api/auth/verify-phone	Phone verification

PROPERTIES

Method	Endpoint	Description
GET	/api/properties	Get all properties (with filters)
GET	/api/properties/:id	Get single property details
POST	/api/properties	Create new property (owner only)
PUT	/api/properties/:id	Update property (owner only)
DELETE	/api/properties/:id	Delete property (owner only)

SEARCH & FILTERS

Method	Endpoint	Description
GET	/api/properties/search	Advanced search
GET	/api/properties/filter?location=x&budget=y	Filter results

IMAGES

Method	Endpoint	Description
POST	/api/properties/:id/images	Upload images
GET	/api/properties/:id/images	Get all images for property

REVIEWS

Method	Endpoint	Description
POST	/api/reviews	Create review
GET	/api/reviews/:property_id	Get reviews for property

INQUIRIES/MESSAGES

Method	Endpoint	Description
POST	/api/inquiries	Send inquiry
GET	/api/inquiries	Get user inquiries
PUT	/api/inquiries/:id	Reply to inquiry

WISHLIST

Method	Endpoint	Description
POST	/api/wishlist	Add to wishlist
GET	/api/wishlist	Get user wishlist
DELETE	/api/wishlist/:id	Remove from wishlist

USER

Method	Endpoint	Description
GET	/api/users/:id	Get user profile
PUT	/api/users/:id	Update user profile
GET	/api/users/:id/listings	Get owner's properties (dashboard)

ANALYTICS

Method	Endpoint	Description
GET	/api/analytics/owner	Owner dashboard stats
GET	/api/analytics/price-trends	Price trends for city

11. QUICK START SETUP COMMANDS

```
# Clone or create project
git init RoomGi && cd RoomGi

# Frontend Setup
npx create-react-app frontend
cd frontend
npm install axios react-query react-router-dom tailwindcss leaflet mapillary react-icons
formik yup
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
cd ..

# Backend Setup
mkdir backend && cd backend
npm init -y
npm install express cors dotenv sequelize pg bcryptjs jsonwebtoken multer nodemailer
cloudinary
npm install -D nodemon
cd ..

# Database Setup (PostgreSQL)
# Install PostgreSQL locally or use Docker:
# docker run --name roomgi-db -e POSTGRES_PASSWORD=password -p 5432:5432 -d postgres

# Create .env files:
# backend/.env:
# DB_HOST=localhost
# DB_USER=postgres
# DB_PASSWORD=password
# DB_NAME=roomgi
# JWT_SECRET=your-secret-key
# CLOUDINARY_KEY=your-key
# SENDGRID_KEY=your-key

# Start development
# Terminal 1: cd backend && npm run dev
# Terminal 2: cd frontend && npm start
```

12. WINNING TIPS

Do's

- **Combine all 3 problem statements** into one unified platform (unique angle)
- **Mobile-first responsive design** (most users browse on phones)
- **Mapillary integration** with street view (wow factor!)
- **Real verification system** (differentiator from OLX)
- **Working analytics dashboard** (shows maturity)
- **Price predictor/comparison** (AI wow factor)
- **Fast & clean UI** (Tailwind makes this easy)
- **Deploy on Vercel + Render** (judges can access live link)

- **GitHub README that's professional** (first impression)

Don'ts ❌

- Don't build only landing pages (judges want functional website)
- Don't hardcode data (use real database)
- Don't ignore mobile experience
- Don't forget error handling
- Don't make confusing navigation
- Don't deploy with placeholder data only
- Don't skip testing
- Don't ignore security (hash passwords, validate inputs, use HTTPS)

> **Judging Psychology** 💭
>
> Judges look for: **Working product > Pretty design > Code quality > Innovation**
>
> - Make it WORK first (even with dummy data)
> - Make it PRETTY second (Tailwind + clean layout)
> - Make it STABLE third (no console errors)
> - Make it UNIQUE fourth (Mapillary + AI = differentiator)

13. RESOURCES TO DOWNLOAD

Resource	Link	Why
Figma Templates	figma.com/community	UI inspiration
React Components	tailwindui.com	Copy-paste components
Icons	react-icons.github.io	10k+ free icons

Sample Data	mockaroo.com	Generate dummy property data
PostgreSQL GUI	pgadmin.org	Easy database management
Postman	postman.com	Test APIs during development
VSCode Extensions	ES7+, Tailwind CSS IntelliSense, REST Client	Speed up coding

14. TIMELINE ESTIMATE (For 2-3 Person Team)

Phase	Duration	Tasks
Planning & Setup	6 hours	Design, DB schema, API planning, environment setup
Backend Core	12 hours	Auth, CRUD APIs, database queries, basic testing
Frontend Core	12 hours	Pages, components, routing, basic styling
Integration	8 hours	Connect frontend to backend, test APIs
Features	10 hours	Mapillary, filtering, search, messaging

Polish & Deploy	8 hours	Mobile responsive, bug fixes, deploy to Vercel/Render
Presentation	2 hours	Create slides, demo script, walkthrough
Buffer	6 hours	Unexpected issues, performance fixes
TOTAL	**~64 hours**	**~26 hours/person for 2-3 weeks part-time**

15. FINAL CHECKLIST BEFORE SUBMISSION

- All 3 problem statements covered (rental + buy-sell + owner dashboard)
- Website is fully functional (not just landing page)
- Responsive on mobile, tablet, desktop
- Search & filtering working
- Owner dashboard CRUD working
- Mapillary integration working
- Images upload working
- Messages/inquiries working
- Reviews & ratings working
- Database has real schema (not just JSON)
- Deployed to Vercel (frontend) + Render (backend)
- GitHub repository is public with good README
- No console errors or warnings
- API endpoints tested with Postman
- Password hashing implemented
- CORS configured correctly
- Environment variables not exposed
- Code is commented (especially complex logic)
- Performance: pages load < 3 seconds
- Presentation slide deck ready (10-15 slides)

16. PRESENTATION STRUCTURE (10 Minutes)

Slide	Content	Duration
Slide 1-2	**Problem Statement:** Show fragmented rental discovery problem, current pain points (fake listings, no transparency)	1 min
Slide 3-4	**Your Solution:** Unified platform for 3 user types, key differentiators (Mapillary, verification, price prediction)	1 min
Slide 5-6	**Demo (LIVE if possible):** Show search functionality, owner dashboard, messaging	4 min
Slide 7-8	**Technical Stack:** Frontend: React + Tailwind, Backend: Node.js + PostgreSQL, APIs: Mapillary, OpenStreetMap	1 min
Slide 9	**Unique Features:** Street view integration, price predictor, verification badges, analytics dashboard	1 min
Slide 10	**Scalability & Future:** Can handle 1M+ listings, ready for mobile app, AI chatbot	1 min
Slide 11	**Team & Timeline:** Team members, built in 48 hours	30 sec
Slide 12	**Call to Action:** "RoomGi: Making real estate transparent & trustworthy"	30 sec

---

🚀 YOU'VE GOT THIS!

Remember:

- ✓ Start simple, add complexity gradually
- ✓ Get something working ASAP (don't perfect first)
- ✓ Judges value working MVP over perfect concept
- ✓ Combine all 3 problems = 10x differentiation
- ✓ Mapillary + price predictor = wow factor
- ✓ Deploy early, troubleshoot live if needed

Good luck! 🏆

Last Updated: January 27, 2026

Made for hackathon champions 🚀