THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCES

NAME: LUKUNDO NAKAMBA

COMPUTER NUMBER: 2022077385

COURSE NAME: SOFTWARE ENGINEERING

COURSE CODE: 3600

Community Board of Realtors

1. Component Identification table

component	responsibility
ListingManager	Creates/validates/stores listings
SearchEngine	Handles search and view operations for listings
ReportGenerator	Creates weekly listing books and other reports
NotificationService	Sends notifications about listing changes
UserAuth	Handles user authentication and authorization
StatusTracker	Manages listing status changes and updates
ScheduleManager	Schedules appointments

2. Interface Design (3 Selected Components)

ListingManager

Methods:

addListing(propertyDetails: Property): boolean

updateListing(listingId: string, newDetails: Property): boolean

deleteListing(listingId: string): boolean

getListing(listingId: string): Property

Dependencies: DatabaseConnector, ImageHandler

SearchEngine

Methods:

searchByLocation(location: string): Property[]

filterByPrice(min: number, max: number): Property[]

sortByDate(ascending: boolean): Property[]

Dependencies: DatabaseConnector, ListingManager

UserAuth

Methods:

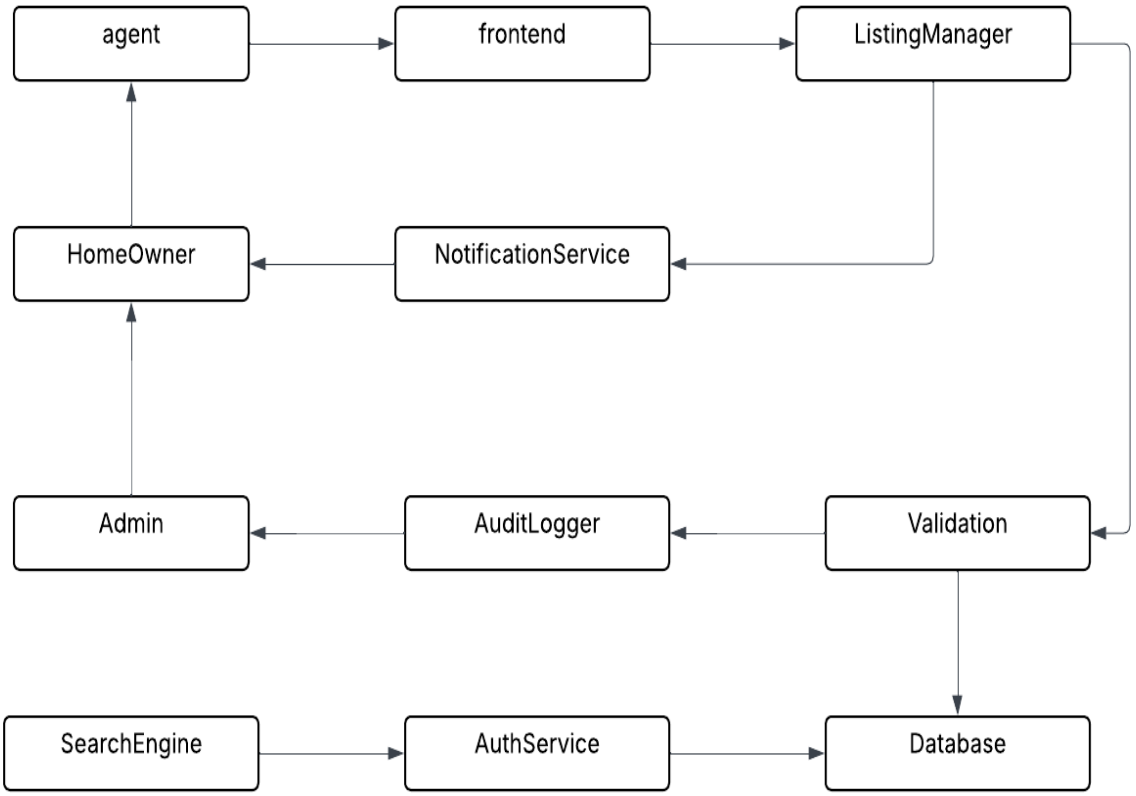
login(username: string, password: string): SessionToken

register(userData: User): boolean

checkRole(userId: string): Role

Dependencies: DatabaseConnector, NotificationService

3. Component Diagrams



4. Iteration Plan

Iteration 1: Core Infrastructure (Weeks 1-2)

Objective: Establish secure foundation for listing creation
Components:

- **ListingManager** (Core logic)
- **Database** (PostgreSQL)
- **AuthService** (RBAC)

Risk Analysis:

The highest-risk scenario is unauthorized data access or corruption during initial listing creation. Without proper authentication (AuthService), malicious actors could inject invalid data, while database failures (Database) might cause irreversible data loss. We mitigate this by implementing JWT authentication with IP restrictions and atomic database transactions that roll back on failure.

Justification:
Security and data integrity risks must be addressed before adding business logic. Implements "fail-secure" foundation.

Iteration 2: Quality Gate (Weeks 3-4)

Objective: Ensure listing validity and stakeholder awareness
Components:

- **ValidationEngine**
- **NotificationService**
- **AuditLogger** (Basic)

Risk Analysis:

The primary risk shifts to data quality invalid listings (e.g., incorrect pricing) could erode user trust or violate regulations. ValidationEngine addresses this by enforcing 25+ business rules (e.g., price/sqft thresholds), while NotificationService reduces communication gaps by alerting homeowners immediately. A secondary risk of over-notification is mitigated with rate limiting. This phase delivers tangible value by preventing support tickets caused by bad data.

Justification:
Prevents garbage-in/garbage-out scenarios while meeting legal notification requirements.

Iteration 3: Compliance & Visibility (Weeks 5-6)

Objective: Enable oversight and discovery
Components:

- **AuditLogger** (Full)

- **SearchEngine** (Basic index)
- **NotificationService** (Enhancements)

Risk Analysis:

Legal exposure becomes the critical concern without immutable audit logs (AuditLogger), disputes over listing changes could lead to liability. Cryptographic hashing of logs ensures tamper-proof records. Meanwhile, SearchEngine's basic indexing, though limited, mitigates future scalability risks by avoiding a costly rearchitecture later. The business gains both legal protection and a foundation for growth.

Justification:
Completes legal compliance requirements while laying groundwork for future features.

The Spring Breaks ‘R’ Us Travel Service

1. Component Identification table

Component Name	Responsibility	Related Use Cases
ResortBrowser	Display available resorts with filters	Browse Resorts, Book Trip
BookingManager	Handle reservation creation/modification	Book Trip, Modify Group, Cancel Booking
PaymentProcessor	Manage payment transactions	Make Final Payment, Send Final Payment Notice
NotificationService	Send booking confirmations/payment reminders	Send Final Payment Notice, Book Trip
UserProfileManager	Store guest details/preferences	Book Trip, Modify Group
InventoryManager	Track room availability in real-time	Book Trip, Change Room Type

2. Interface Design (Examples)

BookingManager Component

- **Methods**:
    - createBooking(guestId, resortId, dates): BookingConfirmation
    - modifyBooking(bookingId, newDates): Status
    - cancelBooking(bookingId): RefundAmount
- **Dependencies**: InventoryManager (check room availability), PaymentProcessor (process refunds).

PaymentProcessor Component

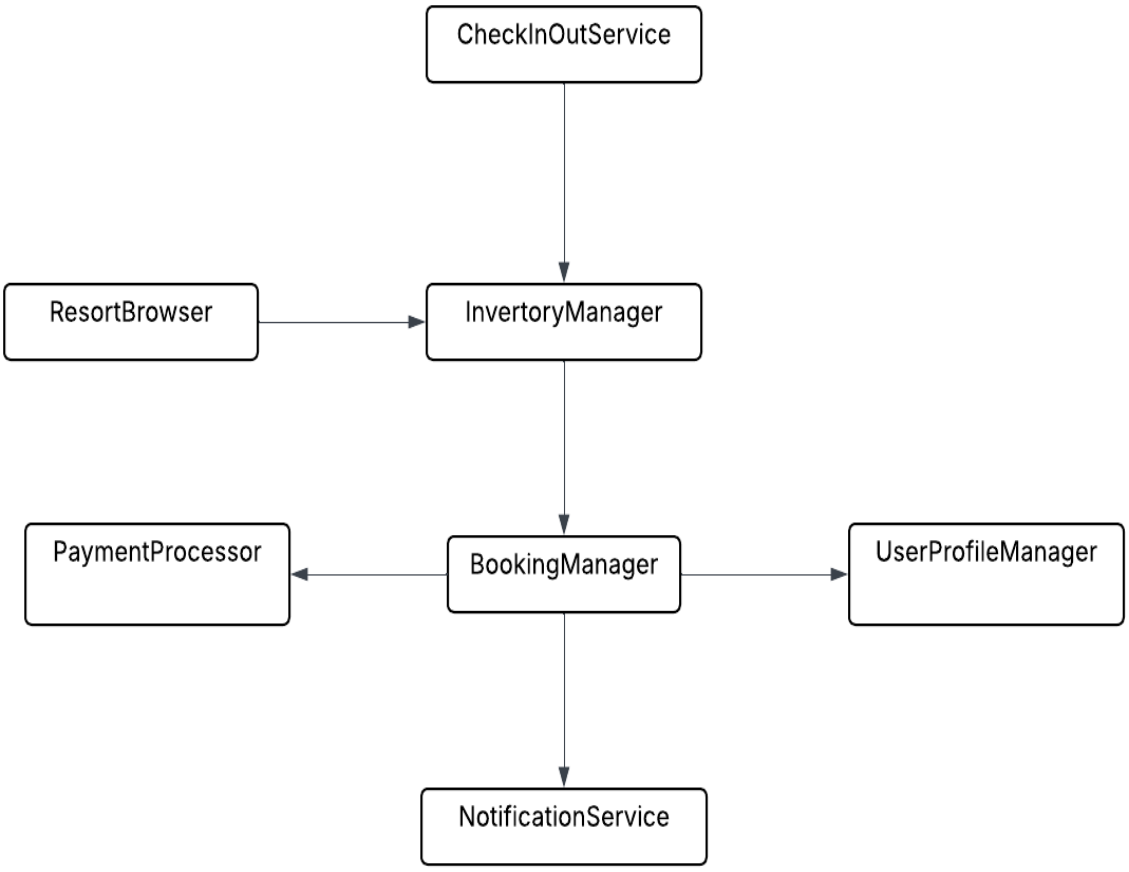
- **Methods**:

    - processPayment(bookingId, cardDetails): TransactionReceipt
    - sendPaymentLink(email): NotificationStatus
- **Data Formats**: JSON for payment requests (e.g., { "bookingId": "123", "amount": 500 }).

NotificationService Component

- **Methods**:
    - sendEmail(userId, templateId): DeliveryStatus
    - sendSMS(phoneNumber, message): DeliveryStatus
- **Dependencies**: External SMTP/SMS gateway.

3. Component Diagrams



4. Iteration Plan (Risk-Based for "Book Trip")

Iteration	Components	Risk Justification
1	ResortBrowser, InventoryManager	High risk: Real-time inventory sync is critical to avoid overbooking.
2	BookingManager, UserProfileManager	Medium risk: Complex business logic for reservations.
3	PaymentProcessor, NotificationService	Low risk: Can use stubs initially; integrate payment gateways last.

Risks Mitigated:

- Iteration 1 addresses data consistency early.
- Iteration 2 ensures core booking workflows are stable.
- Iteration 3 delays external dependencies (e.g., payment APIs).