System Design Specification

for

Project Title: UzBnB

Project Code: 2F-uzbnb

Version: 0.0.7

Date: 24/12/2023

Contents

1 Document Management

1.1 Contributors

Role	Unit	Name
Systems Analyst Designer (Owner)		Komiljon Qosimov
Business Analyst		
Project Manager		Ismoil Salohiddinov
Project Sponsor		
Other document contributors		

1.2 Version Control

Please document all changes made to this document since initial distribution.

Date	Version	Author	Section	Amendment
01/12/2023	0.0.1	Komiljon	Introduction	Initial document release
04/12/2023	0.0.2	Ismoil	Section 3, 8	Added implementation details
05/12/2023	0.0.3	Komiljon	Section 7	Implemented database structure
08/12/2023	0.0.4	Ismoil	Section 6	Added application security
13/12/2023	0.0.5	Ismoil	Section 4, 5, 2	Updated glossary
20/12/2023	0.0.6	Komiljon	Section 9, 10.4	Added data management
24/12/2023	0.0.7	Ismoil	Section 2	Updated glossary

2 OVERVIEW

2.1 General Overview

The system design of UzBnB aims to provide a comprehensive and user-friendly platform for property rentals, offering both guests and hosts a seamless experience. This section provides a high-level overview of the system's functionality and outlines the technical implementation approach.

2.2 Key Features

2.2.1 User Management

The system allows users to create accounts, manage profiles, and authenticate securely. User roles include guests and hosts, each with distinct functionalities.

2.2.2 Property Listings

Hosts can list their properties, providing details such as address, description, and availability. Guests can search, view, and book properties based on their preferences.

2.2.3 Reviews and Ratings

Users can submit reviews and ratings for properties they have experienced, enhancing transparency and trust within the UzBnB community.

2.2.4 Booking and Payment

The system supports a secure booking process, including payment integration. Guests can initiate bookings, and hosts can manage reservations through an intuitive interface.

2.3 Technical Implementation Approach

2.3.1 Architecture

UzBnB follows a three-tier architecture: presentation layer, application layer, and data layer. The presentation layer includes web interfaces for guests and hosts. The application layer handles business logic and includes controllers for user, property, review, and booking management. The data layer encompasses the database storing user details, property information, reviews, and transactions.

2.3.2 Technologies

- **Frontend:** HTML, CSS, JavaScript, ReactJS
- **Backend:** Node.js, Express.js
- **Database:** MongoDB
- **Payment Integration:** [Payment Gateway Provider]
- **Version Control:** Git

2.3.3 Security

Security measures include secure user authentication, data encryption for sensitive information, and regular security audits. HTTPS is enforced for data transmission, and user permissions are strictly managed based on roles.

2.3.4 Scalability

The system is designed to scale horizontally by adding more servers to handle increased user loads. Database sharding and caching mechanisms ensure optimal performance during peak times.

3 DEVELOPMENT TOOLS AND STANDARDS

3.1 Development Tools

- **Frontend:**
  - **HTML, CSS, JavaScript:** Standard web technologies for structuring content, styling, and client-side scripting.
  - **ReactJS:** A JavaScript library for building user interfaces, providing a responsive and dynamic user experience.
- **Backend:**
  - **Node.js:** A JavaScript runtime for server-side development, ensuring consistent language usage throughout the application.
  - **Express.js:** A web application framework for Node.js, simplifying the development of robust and scalable APIs.
- **Database:**
  - **MongoDB:** A NoSQL database for storing user details, property information, reviews, and transactions. Its flexible schema and scalability are well-suited for UzBnB's data model.
- **Version Control:**
  - **Git:** A distributed version control system for tracking changes, managing collaboration, and ensuring a stable codebase.

3.2 Development Standards

Standard	**√ indicates compliance**
Database Design	√ Follows best practices for MongoDB database design, including proper indexing and data modeling.
ReactJS	√ Utilizes ReactJS for building the frontend, providing a responsive and dynamic user interface.
Node.js	√ Implements Node.js for the backend, ensuring a consistent and efficient server-side runtime.
Accessibility	√ Adheres to WCAG standards to ensure the application is accessible to users with disabilities.
Web Style Standards	√ Follows a consistent and visually appealing web style, incorporating responsive design principles.
Supported Web Browsers	√ Compatible with major web browsers, including Chrome, Firefox, and Safari.

4 SYSTEM PROCESSES N/A

This section will be added in future versions of the document.

5 USER INTERFACE N/A

This section will be added in future versions of the document.

6 APPLICATION SECURITY

This section addresses application security considerations, focusing on authentication, authorization, and business object security.

6.1 Authentication

Authentication ensures that users are who they claim to be before granting access to the system.

Implementation:

- Utilize a secure authentication mechanism such as email/password, OAuth, or a multi-factor authentication (MFA) system.
- Encrypt and store user credentials securely.
- Implement secure password policies, including minimum length and complexity requirements.

6.2 Authorization

Authorization controls access to different functionalities, menus, and pages within the application based on user roles.

Implementation:

- Define application roles such as Guest, User, Host, and Administrator.
- Implement role-based access control (RBAC) for menu and page authorizations.
- Enforce least privilege principles, granting users access only to functionalities essential for their roles.

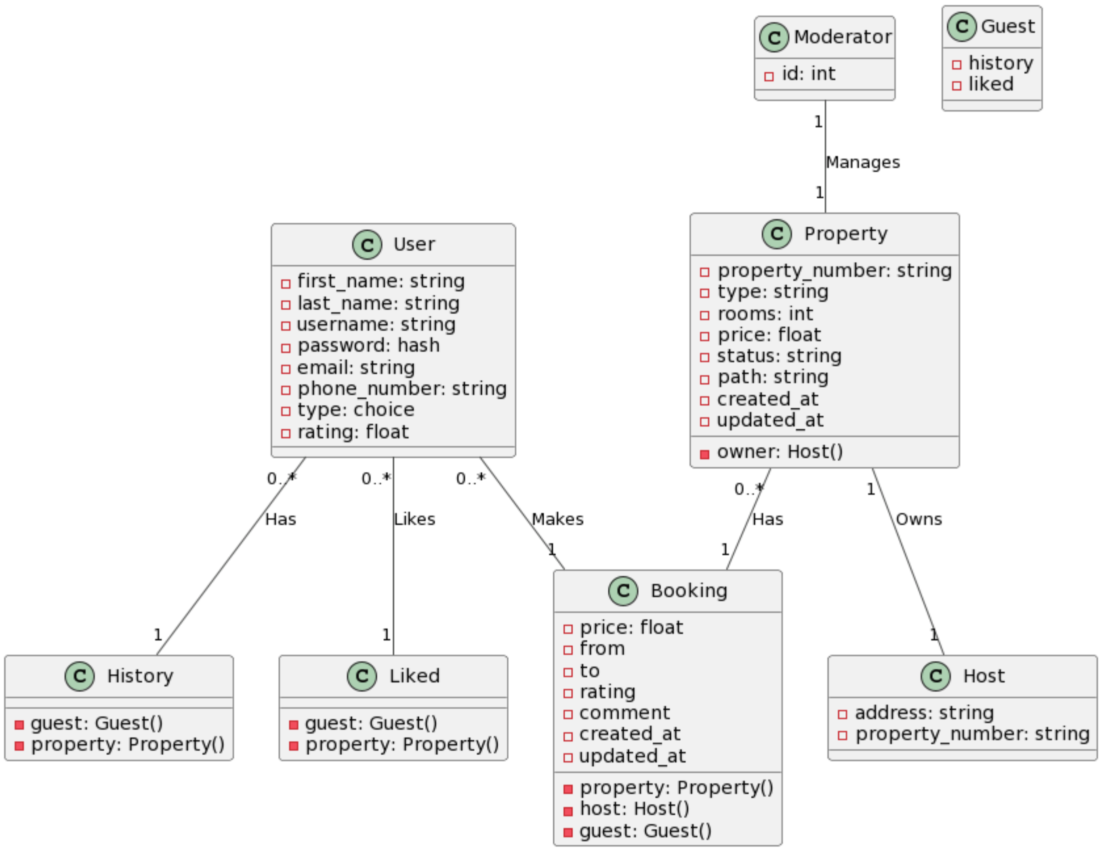
6.3 Business Objects

Business object security ensures appropriate access control to specific objects or rows of data within the application.

Implementation:

- Implement object-level and row-level authorization for sensitive business objects.
- Define access rules based on user roles to restrict or grant permissions to specific data.
- Regularly review and update business object security policies to align with evolving business requirements.

7 DATABASE DESIGN



8 APPLICATION INTERFACES

8.1 Inputs to the System from Other Applications

8.1.1 Property Listing Data Import

- Source Application: Real Estate Database System
- Data Specifications:
  - Data Format: CSV
  - Fields: property_number, address, type, rooms, price, status, path, created_at, updated_at
- Integration Approach: Regular CSV file imports initiated by a scheduled job or triggered by an event in the Real Estate Database System.
- Possible Impact: Improved property listing accuracy and currency.

8.1.2 User Registration and Authentication

- Source Application: Centralized User Management System
- Data Specifications:
  - Data Format: JSON
  - Fields: user_id, username, email, password_hash, user_type, created_at
- Integration Approach: API-based communication for user registration and authentication.
- Possible Impact: Streamlined user onboarding with centralized authentication.

8.2 Outputs from the System to Other Applications

8.2.1 Booking Confirmation Notification

- Destination Application: Email Service
- Data Specifications:
  - Data Format: Plain Text or HTML
  - Fields: recipient_email, subject, message_body
- Integration Approach: Triggered when a booking is confirmed, sending an email notification to the guest.
- Possible Impact: Enhanced communication with users for booking-related updates.

8.2.2 Property Rating and Review Submission

- Destination Application: Marketing Analytics Platform
- Data Specifications:
  - Data Format: JSON
  - Fields: property_id, user_id, rating, comment, submission_timestamp
- Integration Approach: API-based communication to submit property ratings and reviews to the analytics platform.
- Possible Impact: Improved understanding of property performance through analytics.

9 DATA

9.1 Archiving Policy

9.1.1 Archiving Criteria

- Properties: Archive properties that have been inactive or not booked for a specified period.
- Users: Archive user accounts that have been inactive for an extended period.
- Bookings: Archive completed or canceled bookings after a certain retention period.
- Reviews: Archive reviews that are not associated with active properties or users.

9.1.2 Archiving Process

- Define a scheduled job or process to identify and archive records based on the specified criteria.
- Ensure that the archiving process is reversible for compliance and audit purposes.
- Store archived data in a secure and easily retrievable format.

9.1.3 Retention Period

- Define the retention periods in collaboration with relevant stakeholders, considering legal, business, and regulatory requirements.

9.1.4 Impact on Operations

- Archiving should have minimal impact on day-to-day operations.
- Implement proper notifications and logging to track the archiving process.

9.1.5 Data Restoration

- Develop a process for restoring archived data when necessary.
- Ensure that data restoration processes comply with data protection and privacy regulations.

9.1.6 Compliance

- Regularly review and update archiving policies to ensure compliance with changing regulations and business needs.

10 IMPLEMENTATION

When creating an implementation plan at the Build stage for the UzBnB system, several critical implementation issues should be considered to ensure a smooth and successful deployment. Here are key factors to address

10.1 Technology Stack Compatibility

- Verify that the selected technologies (programming languages, frameworks, databases) are compatible with each other and meet the system requirements.
- Ensure that the development and production environments are consistent.

10.2 Scalability

- Consider the potential growth of the user base and property listings.
- Implement scalability measures, such as load balancing and efficient database indexing, to accommodate increased traffic.

10.3 Security Measures

- Implement security best practices throughout the development process.
- Conduct regular security audits and testing to identify and address vulnerabilities.

10.4 Data Privacy and Compliance

- Ensure compliance with data protection regulations (e.g., GDPR, HIPAA) by incorporating privacy measures into the system design.
- Implement user consent mechanisms for data processing activities.

10.5 User Authentication and Authorization

- Implement secure authentication mechanisms (e.g., OAuth) and robust password storage.
- Define and enforce role-based access control to restrict user access based on their roles.

10.6 Error Handling and Logging

- Implement comprehensive error handling mechanisms to capture and log errors.
- Set up logs for monitoring and debugging during the initial stages of deployment.

10.7 Data Backup and Recovery

- Establish regular data backup procedures to prevent data loss.
- Test data recovery processes to ensure a quick and effective response in the event of data corruption or system failure.

10.8 User Training and Documentation

- Develop user documentation and training materials to assist users in navigating and utilizing the system effectively.
- Conduct user training sessions to familiarize stakeholders with system functionalities.

10.9 Performance Testing

- Perform thorough performance testing to identify and address bottlenecks.
- Simulate scenarios with varying levels of user activity to gauge system performance under different conditions.

10.10 Cross-Browser and Cross-Platform Compatibility

- Ensure the system is compatible with major web browsers (e.g., Chrome, Firefox, Safari) and various devices (desktop, tablet, mobile).

- Conduct cross-browser and cross-platform testing to verify consistent functionality.

10.11 Integration with External Systems

- Validate integration points with external systems, such as user authentication or property data imports.
- Ensure seamless data flow between the UzBnB system and other applications.

10.12 Rollback Plan

- Develop a rollback plan in case of unforeseen issues during deployment.
- Test the rollback plan in a controlled environment to ensure its effectiveness.

10.13 Collaboration and Communication

- Foster open communication and collaboration between development, operations, and business teams.
- Establish clear lines of communication for issue reporting and resolution.

Addressing these implementation issues during the Build stage will contribute to a successful deployment of the UzBnB system, reducing the likelihood of post-deployment issues and ensuring a positive user experience.