Software Requirents Specification
(SRS)
TO-LET FINDER

Authors:

Shoaib Hossain (Id-221-15-5949)

Shafiqul Islam (Id-221-15-5945)

Instructor: Lamiya Islam

Lecturer, Department Of CSE

Daffodil International University

1. Introduction:

➢ An online To-let finder will provide the information about family room, bachelor room, Hotel room, Store room, showroom etc.

➢ To-Let Finder application will make easy to find the location of family room, bachelor room, Hotel room, Store room, showroom, select No. of rooms and other facts by the renter.

➢ It will make eassy to upload the location, contract No., expected rent number of rooms facilities and other information by landlord/room master.

➢ This application will be able to show the family room, bachelor room, Hotel room, Store room, showroom with in a particular area selected by User.

(i) Purpose of this document

The purpose of the To-Let Finder Application is to simplify and streamline the process of finding and booking rental properties, catering to the diverse needs of property seekers and owners.

Firstly, it serves as a convenient and centralized platform for property seekers to access comprehensive information about various property types, including family rooms, bachelor rooms, hotel rooms, store rooms, and show rooms. This empowers users to make informed decisions when searching for their ideal rental space.

Secondly, the application facilitates easy booking, enhancing the efficiency of the rental process. It simplifies the booking workflow, making it accessible for tenants and property owners alike. This streamlining of transactions benefits both parties.

Moreover, the application's purpose extends to providing a user-friendly interface, ensuring a seamless and enjoyable experience for users. It aims to save time and effort by offering advanced search and filtering options, allowing users to quickly find properties that align with their preferences and needs.

(ii) Scope of this document

In the future, the To-Let Finder Application can expand its reach by including more property types, diversifying into vacation rentals and commercial spaces. Enhanced user experience with advanced search algorithms, IoT integration for smart properties, and mobile apps can attract a broader audience. Features like multi-lingual support, AI-powered chatbots, and blockchain for security can add value. Collaborations with local businesses and sustainability-focused initiatives can foster partnerships. By staying compliant with evolving regulations and continuously innovating, the application can solidify its position as a leading platform in the property rental industry.

(iii) Definitions :

To-Let Application: The software system being developed to facilitate property rental transactions.

User: Any individual who interacts with the To-Let application, including property owners, tenants, and administrators.

Property Owner: An individual or entity that lists properties for rent on the application.

Tenant: An individual seeking rental properties using the application.

Administrator: A user with privileged access to manage and oversee the application.

Listing: A property advertisement created by a property owner for potential tenants to view**.**

Dashboard: The user interface that allows property owners and tenants to manage their profiles and listings.

Search Functionality: The ability for tenants to search and filter rental properties based on various criteria.

Booking: The process of a tenant expressing interest in renting a property and the property owner confirming the transaction.

2. General Description

The To-Let Finder Application is a sophisticated digital platform designed to revolutionize the property rental experience. This application caters to various property types, including family rooms, bachelor rooms, hotel rooms, store rooms, and show rooms, offering an extensive range of options for property seekers. It is meticulously crafted to provide an intuitive and user-friendly interface, making property search, booking, and location discovery effortless.

2.1. User-Centric Interface

At the heart of this application is a user-centric interface that prioritizes ease of use and accessibility. The platform offers an inviting and visually appealing design, ensuring that property seekers, property owners, and administrators can interact seamlessly with the application.

2.2. Comprehensive Property Listings

The To-Let Finder Application boasts a diverse collection of property listings, catering to various preferences and needs. Property owners can efficiently create, manage, and present their rental properties, with each listing offering extensive details about the property's features, location, pricing, and availability.

2.3. Powerful Search and Discovery

Property seekers have access to robust search and discovery tools, enabling them to fine-tune their property search based on specific criteria. Whether it's location preferences, property type, price range, or desired amenities, the application simplifies the process of finding the perfect rental property.

2.4. User Profiles for Personalization

Enhancing the overall user experience, the To-Let Finder Application provides user profile management. Registered users can personalize their profiles, enabling them to save favorite listings, track booking history, and engage with property owners and fellow tenants.

2.5. Seamless Booking Process

The application offers a secure and hassle-free booking process. Tenants can express their interest in a property and initiate bookings with ease. Property owners, in turn, can efficiently review and confirm booking requests, streamlining the rental agreement process.

2.6. Administrative Empowerment

Administrators are equipped with a suite of powerful tools to maintain the platform's integrity. These tools empower administrators to monitor user activities, address disputes, and ensure the To-Let Finder Application operates efficiently.

3. Functional Requirements

In this section, we will detail the functional requirements of the To-Let application, specifying the features and functionalities that the application must possess to meet the needs of its users.

3.1. User Registration and Authentication

3.1.1. User Registration

User Registration: Users (both property owners and tenants) must be able to register for an account by providing necessary information, including name, email, password, and contact details.

Email Verification: A verification email should be sent to the user's provided email address for account activation.

Social Media Integration: Users should have the option to register and log in using their social media accounts (e.g., Google, Facebook).

3.1.2. User Authentication

User Login: Registered users should be able to log in securely using their email and password or social media credentials.

Forgot Password: Users should have the option to reset their password via email if they forget it.

3.2. Property Listing Management

3.2.1. Property Listing Creation

Property Listing Creation: Property owners should be able to create detailed property listings, including property type, location, price, description, and images.

Multiple Property Listings: Property owners should be able to list multiple properties under their account.

3.2.2. Property Information

Property Details: Each property listing should include essential details such as the number of bedrooms, bathrooms, square footage, amenities, and available date.

Media Upload: Property owners should be able to upload high-quality images and videos of their properties.

Edit and Delete Listings: Property owners should have the ability to edit and delete their property listings.

3.3. Property Search and Filtering

Search Functionality: Tenants should be able to search for rental properties based on location, property type, price range, and other relevant filters.

Sorting: Search results should be sortable by various criteria, such as price, date added, and property type.

Property Details: Users should be able to view detailed property information, including images, descriptions, and contact details for property owners.

3.4. Booking and Transaction Management

Property Booking: Tenants should be able to express interest in a property by requesting a booking. Property owners should have the option to accept or decline booking requests.

Transaction History: Users should be able to view their transaction history, including rental agreements, payment records, and booking status.

Payment Integration: The application should support secure payment processing for rental transactions, including options for credit card payments and digital wallets.

3.5. User Profile Management

User Profiles: Users should have profile pages displaying their contact information, property listings (for property owners), booking history (for tenants), and reviews.

Review and Rating: Users should be able to leave reviews and ratings for properties and other users after a successful transaction.

Profile Editing: Users should be able to edit and update their profile information.

3.6. Administrator Functions

Admin Dashboard: Administrators should have access to a dashboard for managing user accounts, resolving disputes, and overseeing the application's overall functionality.
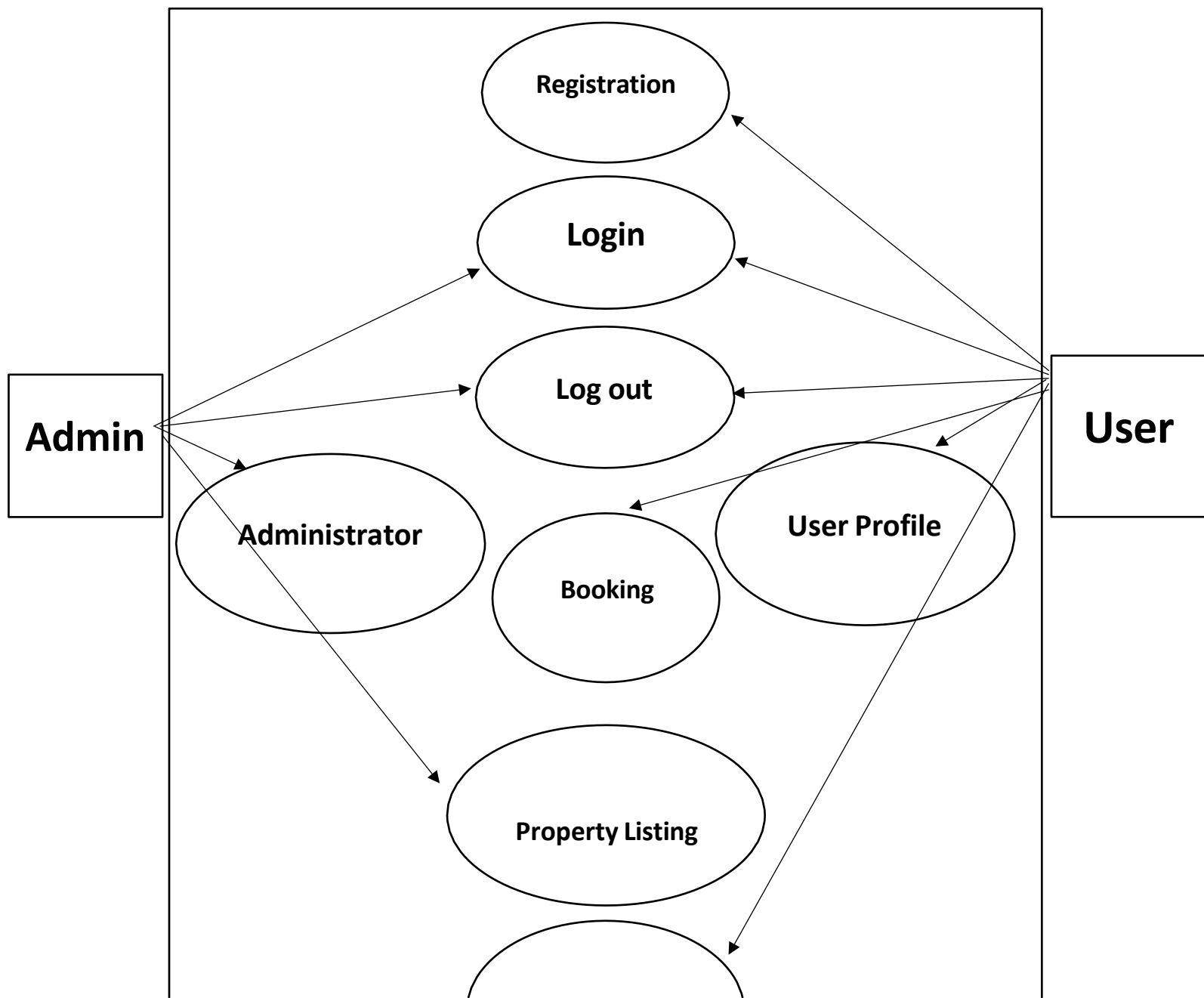
User Management: Admins should be able to suspend or ban user accounts if they violate the application's terms of service or engage in fraudulent activities.
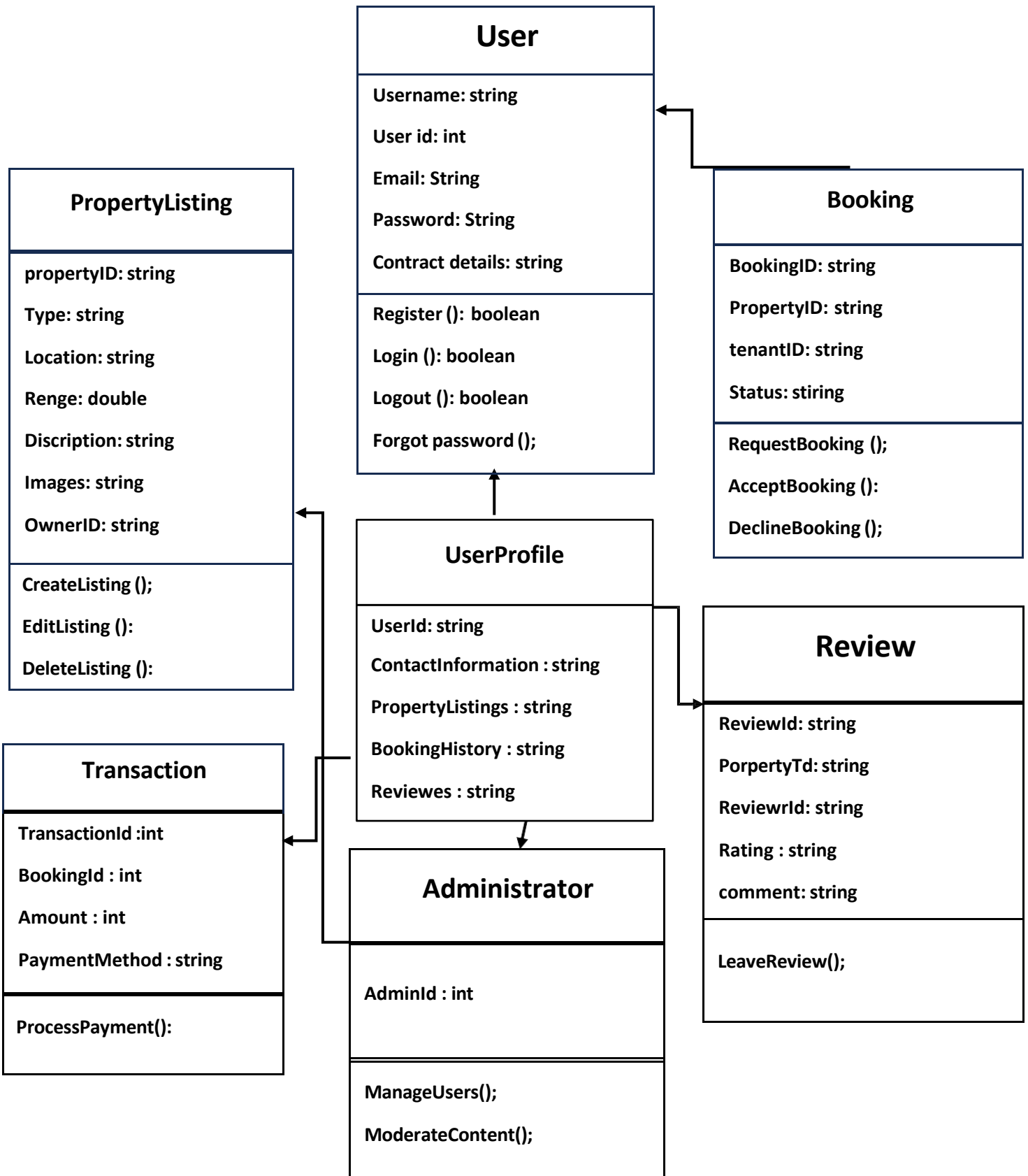
Content Moderation: Admins should have tools for moderating property listings, reviews, and other user-generated content to ensure compliance with community guidelines.

Modeling Requirements

Use Case Diagram :

Registration

Login

Log out

Admin

Administrator

Booking

User Profile

User

Property Listing

Class Diagram :

PropertyListing

propertyID: string

Type: string

Location: string

Renge: double

Discription: string

Images: string

OwnerID: string

CreateListing ();

EditListing ():

DeleteListing ():

User

Username: string

User id: int

Email: String

Password: String

Contract details: string

Register (): boolean

Login (): boolean

Logout (): boolean

Forgot password ();

Booking

BookingID: string

PropertyID: string

tenantID: string

Status: stiring

RequestBooking ();

AcceptBooking ():

DeclineBooking ();

UserProfile

UserId: string

ContactInformation : string

PropertyListings : string

BookingHistory : string

Reviewes : string

Review

ReviewId: string

PorpertyTd: string

ReviewrId: string

Rating : string

comment: string

LeaveReview();

Transaction

TransactionId :int

BookingId : int

Amount : int

PaymentMethod : string

ProcessPayment():

Administrator

AdminId : int

ManageUsers();

ModerateContent();

4. Interface Requirements

The user interfaces of the To-Let Finder Application play a pivotal role in providing an intuitive and engaging user experience. This section outlines the design specifications, wireframes, and interactions between the application and external systems or APIs.

4.1. User-Friendly Interface

Homepage: The application's homepage will feature an uncluttered design with clear navigation options, prominently displaying property categories (family rooms, bachelor rooms, etc.), search functionality, and user login/registration options.

4.2. Mobile Responsiveness

Responsive Design: The application will be designed to be responsive across various devices and screen sizes, ensuring a consistent and user-friendly experience on mobile phones, tablets, and desktops.

4.3. Booking and Reservation Flow

Booking Process: The booking interface will guide users through a seamless booking process, with clear steps, including property selection, date selection, contact information, and payment options.

4.4. Messaging and Communication

In-App Messaging: Users will have the ability to communicate with property owners and fellow tenants through an in-app messaging system. Conversations will be organized and easily accessible.

4.5. Integration with Mapping Services

Location Services: The application will integrate with mapping services (e.g., **Google Maps**) to display property locations accurately. Users can view property locations on maps within the application.

4.6. External Systems and APIs

Third-Party Integration: The application will integrate with external systems or **API**s for features such as **location mapping**, **payment processing**, and **user authentication**. These integrations will be seamless and provide a cohesive user experience.

4.7. Accessibility

Accessibility Standards: The application will adhere to accessibility standards (e.g., **WCAG**) to ensure that all users, including those with disabilities, can access and use the platform effectively.

4.8. User Guidance

Onboarding and Help Sections: The application will include onboarding tutorials and help sections to guide users through the platform's features and functionalities, ensuring a smooth and intuitive user experience.

4.10. Cross-Browser Compatibility

Cross-Browser Compatibility: The application will be compatible with a range of **web browsers**, **including Chrome**, **Firefox, Safari**, and **Edge**, to ensure consistent functionality and appearance.

5. Performance Requirements

The performance of the To-Let Finder Application is crucial to delivering a satisfactory user experience. This section outlines the specific performance criteria that the application must meet, including response times, scalability, and system availability. Additionally, it defines the procedures for performance testing to ensure these requirements are met.

5.1. Response Times

Page Load Time: The average page load time for the application's web pages should not exceed 3 seconds for a broadband internet connection. This ensures that users can access information quickly.

Search Response Time: The search functionality should return results within 1 second of a user's query submission, even during peak usage.

5.2. Scalability

Concurrent Users: The application should be able to handle a minimum of 1,000 concurrent users without significant performance degradation. This ensures that the platform remains responsive during high-demand periods.

5.3. System Availability

5.4. Performance Testing

Load Testing: Load testing will be conducted to assess how the application performs under different levels of user load. This will ensure that it can handle the expected number of users without significant performance issues.

Response Time Testing: Response time testing will measure and analyze the application's response times under various scenarios to ensure they meet the specified criteria.

5.5. Performance Optimization

Caching: Implement caching mechanisms to reduce database queries and improve page load times**.**

Database Indexing: Optimize database performance through proper indexing and query optimization**.**

5.6. Disaster Recovery and Redundancy

Data Backup: Regularly back up user data and application databases to ensure data integrity and availability in case of unforeseen events**.**

6. Design Constraints

6.1. Platform Compatibility

Cross-Browser Compatibility: The application must be compatible with a wide range of web browsers, including but not limited to Chrome, Firefox, Safari, and Edge. Ensuring consistent functionality and appearance across different browsers is imperative.

Mobile Responsiveness: The design must be responsive to varying screen sizes and devices, including smartphones and tablets. Mobile users should have a seamless and user-friendly experience**.**

6.2. Third-Party Integrations

External APIs: The application will integrate with external systems and APIs for services such as mapping, payment processing, and user authentication. The design must accommodate these integrations and ensure smooth data exchange.

Third-Party Services: Any third-party services used, such as mapping services, must adhere to their terms of use and licensing agreements. Legal constraints associated with these services must be considered**.**

7. Non-Functional Attributes

7.1. Reliability

Availability: The To-Let application should have a high level of availability, with a target uptime of at least 99.9%. Planned maintenance and downtime should be communicated to users in advance**.**

Fault Tolerance: The application should be designed to handle and recover gracefully from unexpected errors or system failures without data loss**.**

7.2. Security

Authentication and Authorization: User authentication must be secure, utilizing strong encryption and secure hashing algorithms. Role-based access control should be implemented to ensure that users only access authorized functionality.

Data Encryption: Sensitive user data, such as passwords and payment information, should be encrypted both in transit (using HTTPS) and at rest to prevent unauthorized access.

Data Privacy: The application must comply with relevant data privacy regulations (e.g., GDPR) and should provide mechanisms for users to manage their data and privacy preferences.

Session Management: Secure session management practices should be implemented to prevent session hijacking and ensure that sessions expire after a period of inactivity.

Logging and Auditing: Comprehensive logging of user activities, access, and system events should be maintained for security and audit purposes.

7.3. Usability

User-Friendly Interface: The application should have an intuitive and user-friendly interface, following best practices in user experience (UX) design to ensure ease of use for all users.

Accessibility: The application should be designed and tested to be accessible to users with disabilities, following accessibility standards like WCAG (Web Content Accessibility Guidelines).

Performance Efficiency: The application should provide responsive and fast user interactions, with page load times and response times optimized for a seamless user experience.

7.4. Maintainability

Modularity: The application code should be organized into modular components, making it easier to maintain and extend in the future**.**

Documentation: Comprehensive documentation, including code comments, user manuals, and system architecture documentation, should be maintained to aid in ongoing development and maintenance efforts**.**

Version Control: Source code should be managed using a version control system (Git) to track changes and facilitate collaboration among developers.

Code Quality: Adherence to coding standards and best practices should be enforced to maintain code quality and readability**.**

Testing and Quality Assurance: Regular testing, including unit testing, integration testing, and security testing, should be performed to identify and rectify issues promptly.

8. Preliminary Schedule and Budget

Developing the To-Let Finder Application requires careful planning and resource allocation. This section provides an initial schedule and budget estimation for the project, offering a high-level overview of the expected timeline and costs.

8.1. Preliminary Schedule

Phase 1: Planning and Requirements Gathering (2 weeks)

-> Define project scope and objectives.

-> Create a detailed Software Requirements Specification (SRS).

-> Identify key stakeholders and establish communication channels.

-> Set up project management and collaboration tools.

Phase 2: Design and Architecture (4 weeks)

-> Create wireframes and design mockups for the user interface.

-> Define the technology stack and architecture.

-> Plan the database structure.

-> Prepare a detailed system design document.

Phase 3: Development (10 weeks)

-> Build the core application features, including user registration, property listing, search functionality, and user profiles.

-> Integrate third-party services and APIs (e.g., mapping, payment processing).

-> Implement booking and transaction management.

Phase 4: Testing and Quality Assurance (4 weeks)

-> Conduct unit testing, integration testing, and performance testing.

-> Identify and resolve bugs and issues.

-> Ensure compliance with accessibility standards.

Phase 5: Deployment (2 weeks)

-> Prepare the application for production deployment.

-> Set up hosting infrastructure and configure servers.

-> Deploy the application to a production environment.

-> Implement monitoring and error tracking systems.

Phase 6: Post-Launch Activities (Ongoing)

-> Monitor application performance and user feedback.

-> Address any post-launch issues and optimize performance.

8.2. Preliminary Budget Estimation

Development Team:

-> Project Manager

-> Front-end Developers

-> Back-end Developers

-> UI/UX Designers

-> Quality Assurance Testers

Estimated Budget Breakdown:

Development Team Salaries: $XXX,XXX (Based on team size, experience, and duration of the project).

Infrastructure and Hosting: $X,XXX (Includes server hosting, domain registration, and cloud services).

Third-Party Service Costs: $X,XXX (Costs associated with external APIs and services).

Design and Graphics: $X,XXX (Design and graphics-related expenses, including wireframes and mockups**).**

Testing and Quality Assurance: $X,XXX (Costs for testing tools, testing environments, and QA resources).