

Stay-Finder

**A PROJECT REPORT
for
Mini Project (KCA353)
Session (2024-25)**

Submitted by

**Abhishek Nayak
2300290140008
Abhishek Gupta
2300290140006**

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr. Amit Kumar Gupta**

Professor



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(DECEMBER 2024)

DECLARATION

I hereby declare that the work presented in this report entitled “Stay-Finder”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Abhishek Nayak

Roll. No: 2300290140008

Branch: MCA

Name: Abhishek Gupta

Roll. No: 2300290140006

Branch: MCA

CERTIFICATE

Certified that **Abhishek Nayak 2300290140008, Abhishek Gupta 2300290140008** have carried out the project work having “**Stay-Finder**” (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Dr. Amit Kumar Gupta
Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Stay-Finder

Abhishek Nayak

Abhishek Gupta

ABSTRACT

Stayfinder is a web-based platform inspired by Airbnb, designed to simplify the process of finding and listing properties. This mini-project includes essential modules such as signup and login functionality, property listings, review and rating features, and map integration for enhanced usability. Built using modern web technologies including HTML5, CSS3, JavaScript, Bootstrap, EJS, Node.js, and MongoDB, Stayfinder ensures a seamless user experience and robust performance.

The project aims to replicate core functionalities of property rental systems while focusing on usability and simplicity. Users can register, log in, explore property listings, leave reviews, and view property locations through an interactive map. The development process involved leveraging resources like the Airbnb website, Apna College tutorials, and extensive documentation from MDN, Bootstrap, and other platforms.

Stayfinder demonstrates the effective use of full-stack development principles and highlights the integration of various tools and frameworks to create a functional and user-friendly application. This report outlines the development process, the challenges faced, and the solutions implemented to build this platform.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Dr . Amit Kumar Gupta (Professor) for their guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Abhishek Nayak(2300290140008)

Abhishek Gupta (2300290140006)

TABLE OF CONTENTS

Declaration	ii
Certificate	iii
Abstract	iv
Acknowledgements	v
Table of Contents	vi
1. INTRODUCTION	9-11
1.1 Overview	9
1.2 Description	9
1.3 Key Features	9-10
1.4 Project Vision	10
1.5 Applications of StayFinder	10-11
1.6 Benefits of StayFinder	11
2. FEASIBILITY STUDY	12-14
2.1 Technical Feasibility	12
2.2 Operational Feasibility	13
2.3 Feasibility Study	13-14
3. SYSTEM REQUIREMENTS	15-17
3.1 System Requirements	15
3.2 Hardware Requirements	15-16
3.3 Software Requirements	16
3.4 Network Infrastructure	16-17
3.5 Security Considerations	17
3.5 Scalability	17
4. SYSTEM ANALYSIS	18-24
4.1 Requirement Gathering	18-20
4.2 System Design	20-21

4.3 Data Modelling	21-23
4.4 Feasibility Analysis	23
5. DATABASE DESIGN	24-30
5.1 Software Development Lifecycle	24-25
5.2 Use Case Diagram	26-27
5.3 ER Diagram	27-28
5.4 Data Flow Diagram	28-29
5.5 Flow Diagram	30
6. TESTING	31-37
6.1 Introduction	31
6.2 Testing Strategy and Methodologies	31-32
6.3 Types of Testing Conducted	33-36
6.4 Detailed Test Case for Each Module	34-36
6.5 Results and Analysis	37
7. PROJECT SCREENSHORTS	38-41
7.1 Home Page	38
7.2 Sign In	39
7.3 Login	39
7.4 Create New Listings	40
7.5 Review and Ratings	40
7.6 Map Integration	41
8. MAINTENANCE AND FUTURE SCOPE	42-47
8.1 Maintenance Objectives	42
8.2 Maintenance Strategies	42-43
8.3 Maintenance Process	43-44
8.4 Future Scope	44-47
9. CONCLUSION	48-49

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
5.1	Prototype Model	25
5.2	Use Case Diagram	27
5.3	ER Diagram	28
5.4	Data Flow Diagram	29
5.5	Flow Chart	30
7.1	Home Page	38
7.2	Sign In	39
7.3	Login	39
7.4	Create New Listings	40
7.5	Review and Ratings	40
7.6	Map Integration	41

Chapter 1

INTRODUCTION

1.1 Overview

Stayfinder is a web-based application inspired by Airbnb, offering users a platform to list, search, and book rental properties. It is designed to simplify the process of finding accommodations by providing essential functionalities like user authentication, property management, and location visualization. Stayfinder serves as a one-stop solution for individuals looking for convenient and trustworthy accommodation options.

1.2 Description

Stayfinder addresses the growing demand for online property rental platforms. The system offers seamless interaction for both property owners and users, allowing owners to list their properties and users to book them with ease. This platform leverages advanced technologies like MongoDB for data management, Node.js for server-side scripting, and Google Maps API for location-based services. Its intuitive interface ensures that users, regardless of technical expertise, can navigate the platform effortlessly.

The platform is built to prioritize security, efficiency, and scalability. The authentication system uses encrypted protocols to protect user data, while the database is optimized to handle multiple concurrent transactions. Stayfinder's modular architecture allows easy integration of future enhancements, ensuring the platform remains relevant and adaptable to emerging trends.

1.3 Key features

1. **User Authentication:** Secure login and signup functionality to protect user account.

2. **Property Listings:** Allows property owners to create and manage their property details, including descriptions, images, and pricing.
3. **Search and Filter Options:** Users can search for properties based on various criteria such as location, price range, and property type.
4. **Reviews and Ratings:** Facilitates user feedback on properties, enhancing transparency and trustworthiness.
5. **Map Integration:** Displays property locations dynamically using Google Maps for a better user experience.
6. **Responsive Design:** Ensures the platform works seamlessly across devices, including desktops, tablets, and smartphones.

1.3 Key Features

- To create a user-friendly interface for property rental and listing.
- To implement secure and scalable modules for managing user data and property details.
- To provide a reliable platform for users to book accommodations conveniently.
- To enhance trust and credibility through user reviews and ratings.
- To integrate map-based visualization for improved property location identification.
- To utilize modern web technologies for building a responsive and interactive platform.

1.4 Project Vision

Stayfinder envisions bridging the gap between property owners and users by offering a streamlined, reliable, and secure platform for property rentals. The project aims to enhance the rental experience by leveraging technology to create an efficient and trustworthy ecosystem. Through continuous innovation and improvement, Stayfinder aspires to set a benchmark in the online property rental industry.

1.5 Applications of Stayfinder

1. **Travel and Tourism:** Enables tourists to find affordable and suitable accommodations for their trips.
2. **Real Estate:** Provides property owners a platform to showcase their spaces for short-term rentals.
3. **Business Travel:** Assists professionals in finding convenient stays during work-related travels.
4. **Student Housing:** Offers students affordable and temporary housing options near their institutions.

5. **Event Accommodation:** Facilitates guests in finding stays for weddings, conferences, and other events.

1.6 Benefits of Stayfinder

- **Convenience:** Simplifies the process of finding and booking rental properties.
- **Trustworthiness:** Builds trust through user reviews and ratings for properties.
- **Efficiency:** Saves time with advanced search and filtering options.
- **Scalability:** Supports an increasing number of users and property listings without compromising performance.
- **Enhanced User Experience:** Delivers a responsive, visually appealing interface with intuitive navigation.
- **Global Accessibility:** Allows users to access the platform from anywhere, at any time.

Stayfinder is designed to address the diverse needs of modern travelers and property owners, ensuring a seamless and enriched rental experience for all users.

Chapter 2

FEASIBILITY STUDY

A feasibility study evaluates the viability of a project from multiple perspectives to determine whether it can be successfully implemented. For Stayfinder, the feasibility study focuses on three primary aspects: technical feasibility, operational feasibility, and economic feasibility. These factors collectively ensure the project's sustainability, practicality, and cost-effectiveness.

2.1 Technical feasibility

Technical feasibility assesses whether the project's technical requirements can be met with the available resources and expertise. The Stayfinder project involves the use of the following technologies:

- **Frontend Technologies:** HTML5, CSS3, Bootstrap, and JavaScript for creating a responsive and interactive user interface.
- **Backend Technologies:** Node.js for server-side scripting and EJS for rendering dynamic web pages.
- **Database Management:** MongoDB for efficient data storage and retrieval.
- **Map Integration:** Google Maps API for property location visualization.

Key Considerations :

1. Technology Availability:

- All the required tools and frameworks are readily available as open-source or free-to-use resources.
- The development environment is supported by commonly used software like Visual Studio Code.

2. Expertise and Skills:

- The project team is proficient in full-stack web development, particularly in the MERN (MongoDB, Express, React, Node.js) stack.
- Tutorials, documentation, and community support for each technology are accessible to address potential technical challenges.

3. Scalability:

- MongoDB's non-relational database structure allows for handling large datasets efficiently.
- The modular architecture of Node.js enables seamless integration of new features or upgrades in the future.

2.2 Operational feasibility

Operational feasibility evaluates whether the project's goals align with user needs and whether it can function effectively within its intended environment. For Stayfinder, the primary stakeholders include property owners and users seeking rental accommodations.

Key Considerations:

1. User Requirements:

- The platform's user-friendly design simplifies navigation and interaction for both tech-savvy users and novices.
- Features such as advanced search filters, reviews, and map integration directly address user demands for convenience and reliability.

2. Operational Workflow:

- Property owners can easily list their spaces by providing descriptions, images, and pricing.
- Users can search, filter, and book accommodations in a streamlined manner.
- The review and rating system enhances transparency and trust, encouraging repeat usage.

3. Maintenance and Updates:

- Regular updates can be easily deployed due to the project's modular design.
- Customer support mechanisms, including chatbots or email support, can be incorporated to handle user queries and complaints.

4. Adaptability:

- Stayfinder can adapt to diverse use cases, such as short-term stays, student housing, and business travel accommodations.
- The system's architecture allows for multilingual and multi-currency support, catering to a global audience.

2.3 Feasibility Study

Economic feasibility determines whether the project is financially viable, considering the costs involved and the potential benefits. Stayfinder's economic analysis focuses on development costs, operational expenses, and expected returns.

Key Considerations:

1. Development Costs:

- **Human Resources:** The project team comprises developers skilled in the required technologies, minimizing external hiring costs.
- **Tools and Software:** All tools used, including Node.js, MongoDB, and Bootstrap, are free or open-source, significantly reducing expenses.
- **Infrastructure:** Cloud-based services (e.g., AWS or Heroku) can be used for hosting, offering scalable pricing plans.

2. Operational Costs:

- Hosting and domain expenses are manageable with budget-friendly service providers.
- Minimal overhead costs due to the digital nature of the platform.

3. Revenue Generation:

- Stayfinder can generate revenue through commission-based bookings, subscription plans for property owners, and advertisements.
- The integration of advanced features, such as premium listings or enhanced search visibility, can provide additional income streams.

4. Return on Investment (ROI):

- With a low initial investment and multiple revenue streams, Stayfinder's ROI is projected to be high within the first few years of operation.

Chapter 3

SYSTEM REQUIREMENTS

System requirements are the foundational specifications needed for the successful implementation and operation of a software project. The Stayfinder project has been designed with a clear understanding of its system requirements, encompassing hardware, software, network infrastructure, security, scalability, and system modeling through diagrams.

3.1 System Requirements

System requirements outline the essential components necessary for the development and execution of Stayfinder. These requirements include:

1. Functional Requirements:

- User authentication for secure login and signup.
- Dynamic property listings with images and descriptions.
- Map-based location visualization.
- Review and rating functionality.
- Advanced search filters for properties.

2. Non-functional Requirements:

- Scalability to accommodate increased users and property listings.
- High system availability and reliability.
- Responsive design for cross-device compatibility.
- Secure data transmission and storage.

3.2 Hardware Requirements

The hardware requirements specify the necessary physical components for the project's development and deployment:

1. Development Environment:

- Processor: Intel i5 or equivalent.
- RAM: 8 GB minimum.
- Storage: 256 GB SSD.
- Display: Full HD Monitor.

2. Server Requirements:

- Processor: Intel Xeon or equivalent.
- RAM: 16 GB or higher.

- Storage: 1 TB SSD with RAID configuration.
 - Network: High-speed internet connectivity (1 Gbps or higher).
3. **End-user Devices:**
- Smartphone or PC with modern browsers.
 - Internet connection (minimum 10 Mbps).

3.3 Software Requirements

Software requirements detail the platforms, tools, and technologies used in Stayfinder.

1. **Operating Systems:**
 - Windows 10/11, macOS, or Linux (Ubuntu preferred for development).
2. **Development Tools:**
 - Visual Studio Code for coding.
 - Git for version control.
 - Postman for API testing.
3. **Frameworks and Libraries:**
 - Frontend: HTML5, CSS3, JavaScript, Bootstrap.
 - Backend: Node.js with Express.js.
 - Database: MongoDB.
 - Templates: EJS.
4. **APIs:**
 - Google Maps API for geolocation services.

3.4 Network Infrastructure

The network infrastructure ensures seamless communication between the client and server. Stayfinder's requirements include:

1. **Hosting Platform:**
 - Cloud-based hosting services like AWS, Heroku, or DigitalOcean for scalability.
2. **Load Balancers:**
 - Implemented to distribute traffic efficiently.
3. **Network Security:**
 - SSL certificates for encrypted data transmission.
 - Firewalls to prevent unauthorized access.
4. **Internet Requirements:**
 - A stable high-speed internet connection for developers and end-users.

3.5 Security Considerations

Security is a crucial aspect of Stayfinder to ensure data integrity and user trust. Key considerations include:

1. **Authentication and Authorization:**
 - Use of encrypted passwords (bcrypt hashing).
 - Role-based access control (RBAC) for different user levels.
2. **Data Protection:**
 - Implementing database encryption for sensitive information.
 - Regular backups to prevent data loss.
3. **Secure API Calls:**
 - Validation and sanitization of inputs to prevent SQL/NoSQL injection attacks.
4. **Monitoring and Auditing:**
 - Logs and alerts for suspicious activities.

3.6 Scalability

Scalability ensures the system can handle increasing demands as the user base grows. Stayfinder's design includes:

1. **Horizontal Scaling:**
 - Adding more servers to manage traffic spikes.
2. **Vertical Scaling:**
 - Upgrading existing server resources.
3. **Database Optimization:**
 - Indexing and sharding in MongoDB for efficient queries.
4. **Caching Mechanisms:**
 - Use of Redis or Memcached to store frequently accessed data.

Chapter 4

SYSTEM ANALYSIS

4.1 Requirement gathering

The first and most essential step in the development of the Stayfinder project was to gather and analyze the requirements. This phase is vital to understand what the application needs to achieve and what functionalities should be incorporated. The following are the main requirements for Stayfinder, detailed according to different categories:

1. Functional Requirements:

- **Signup and Login Module:**
 - **User Registration:** The platform should allow users to sign up by providing their basic details, including name, email, and password.
 - **User Authentication:** The login process must verify user credentials securely and grant access to the platform for authenticated users.
 - **Password Management:** Users should have the option to reset or recover their password if forgotten.
 - **Session Management:** Secure management of user sessions to keep users logged in and enable them to access protected routes.
- **Listing Module:**
 - **Property Listings:** Users should be able to browse and view different properties listed by hosts.
 - **Property Search and Filters:** Users must have the ability to search for properties using filters such as location, price range, and property type.
 - **Property Details Page:** Each listing should include detailed information about the property, such as images, description, amenities, and price.
- **Review and Rating Module:**
 - **Review Submission:** Users should be able to leave reviews for properties they have stayed in, detailing their experience.
 - **Rating System:** A rating mechanism should be in place, allowing users to rate their stay on a scale of 1 to 5 stars.

- **View Reviews:** Other users should be able to view past reviews and ratings for properties.
- **Review Moderation:** An admin feature to review and manage inappropriate or offensive content.
- **Map Integration:**
 - **Interactive Map:** A map should be integrated into the site to help users find property locations.
 - **Property Markers:** Properties should be marked on the map, showing their location relative to user searches.
 - **Map Customization:** The map should be responsive and easy to navigate on both desktop and mobile devices.

2. Non-Functional Requirements:

- **Performance:** The website should load quickly and perform smoothly under various load conditions.
- **Security:** Sensitive user data, such as passwords, should be securely stored using encryption techniques. The platform must also be protected against common vulnerabilities like SQL injection and cross-site scripting (XSS).
- **Scalability:** The application must be able to scale with an increasing number of users and data without degrading performance.
- **Usability:** The UI/UX design should be intuitive, user-friendly, and consistent across different pages and devices.
- **Responsiveness:** The website must be fully responsive, adapting seamlessly to various screen sizes, including mobile and desktop.

3. System Requirements:

- **Technical Requirements:**
 - **Frontend Technologies:** HTML5, CSS3, Bootstrap, JavaScript
 - **Backend Technologies:** Node.js, EJS (Embedded JavaScript templates)
 - **Database:** MongoDB for storing user and property data
 - **Code Editor:** Visual Studio Code
- **Software Requirements:**
 - **Node.js** for server-side scripting and backend logic.
 - **MongoDB Atlas** or local MongoDB server for database hosting.
 - **Express.js** for creating a web server and handling routing.
 - **Third-Party APIs:** Integration with mapping services like Google Maps for map functionalities.

4. User Requirements:

- **End-Users:**
 - Users should be able to create an account, log in, browse properties, leave reviews, and interact with the map.
- **Admins:**
 - Admins should have the ability to manage user data, review content, and ensure proper functioning of the platform.

4.2 System design

The system design phase involves creating an architecture for the Stayfinder project. This section focuses on defining the structure of the application, how it functions, and the flow between different modules.

1. Architectural Design: Stayfinder adopts a client-server architecture using a three-tier model:

- **Presentation Layer (Frontend):** Responsible for the user interface (UI) and user experience (UX). Built using HTML5, CSS3, Bootstrap, and JavaScript, this layer includes the sign-up and login forms, property listings, review submission forms, and the interactive map.
- **Application Layer (Backend):** Developed with Node.js and Express.js, this layer handles business logic, user authentication, and communication with the database.
- **Data Layer (Database):** MongoDB stores user details, property listings, and review data. The database schema is designed to maintain relationships between users, properties, and reviews.

2. Design Patterns:

- **MVC Pattern (Model-View-Controller):**
 - **Model:** Represents the data structure, including user and property models in MongoDB.
 - **View:** Refers to EJS templates that display the UI dynamically.
 - **Controller:** Handles the logic behind user requests, including authentication, property listing retrieval, and review submission.

3. UI/UX Design:

- **Homepage:** A clean, welcoming interface displaying featured listings, search bar, and navigation menu.
- **Signup/Login Page:** Simple forms with fields for email, password, and buttons for registration and login.
- **Dashboard and Listings Page:** Displays properties with filters, images, and detailed descriptions.
- **Map Page:** An interactive map that integrates with external mapping services and plots property locations.

- **Review Page:** User-friendly interface for submitting and viewing reviews and ratings.

4. Workflow and Interaction:

- **User Journey:** Users start by registering and logging in. After authentication, they can browse listings, view property details, and interact with the map. Users can submit reviews for properties they have visited and rate their experience.
- **Admin Journey:** Admins can log in, manage user accounts, moderate reviews, and handle other administrative tasks.

4.3 Data modelling

Data modelling is essential for defining the database structure and how data will be stored and accessed. MongoDB's schema-less nature allows for flexibility, but data organization is still necessary to maintain efficiency.

1. Entities and Relationships:

- **User Entity:**
 - Attributes: userID, username, email, passwordHash, role (e.g., user, admin), dateCreated
 - Relationships: One-to-many with Property and Review entities.
- **Property Entity:**
 - Attributes: propertyID, ownerID, title, description, location, price, images, availability, rating
 - Relationships: Many-to-one with User (owner), one-to-many with Review.
- **Review Entity:**
 - Attributes: reviewID, userID, propertyID, rating, comment, dateCreated
 - Relationships: Many-to-one with User and Property.

2. Data Flow Diagram (DFD):

- **Level 0 (High-Level View):** Shows the interaction between the user, web application, and database.
- **Level 1 (Detailed View):** Illustrates data flows within the system, such as user registration, property listing retrieval, review submission, and map queries.

3. ER Diagram (Entity-Relationship Diagram): The ER diagram illustrates the relationships between entities:

- **User** and **Property** have a one-to-many relationship.
- **Property** and **Review** have a one-to-many relationship.

4. Database Schema Design:

- **User Collection:**

```
{
  "userID": "ObjectId",
  "username": "String",
  "email": "String",
  "passwordHash": "String",
  "role": "String",
  "dateCreated": "Date"
}
```

- **Property Collection:**

```
{
  "propertyID": "ObjectId",
  "ownerID": "ObjectId",
  "title": "String",
  "description": "String",
  "location": {
    "type": "Point",
    "coordinates": [longitude, latitude]
  },
  "price": "Number",
  "images": ["Array of Strings"],
  "availability": "Boolean",
  "rating": "Number"
}
```

- **Review Collection:**

```
{
  "reviewID": "ObjectId",
  "userID": "ObjectId",
  "propertyID": "ObjectId",
  "rating": "Number",
}
```

5. Data Validation:

- **Frontend Validation:** JavaScript ensures data entered by users is in the correct format before submission.
- **Backend Validation:** Node.js checks data integrity and authenticity before saving it to the database.

4.4 Feasibility Analysis

Feasibility analysis assesses whether the Stayfinder project is practical and achievable within the given constraints. This section covers four main types of feasibility: technical, operational, financial, and legal.

1. Technical Feasibility:

- **Technologies:** The project uses proven technologies like Node.js, MongoDB, and EJS, which are well-supported and widely adopted. The combination of these technologies ensures the development of a scalable, high-performing application.
- **Integration Capabilities:** APIs for user authentication (e.g., JWT for token-based authentication) and map services (e.g., Google Maps) are easily integrable.
- **Development Environment:** Visual Studio Code is a robust code editor that supports all necessary languages and extensions to streamline development.
- **Resource Availability:** The development team has the required skill set and resources to build.

Chapter 5

DATABASE DESIGN

5.1 Software development life cycle model prototype model

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possibly exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed

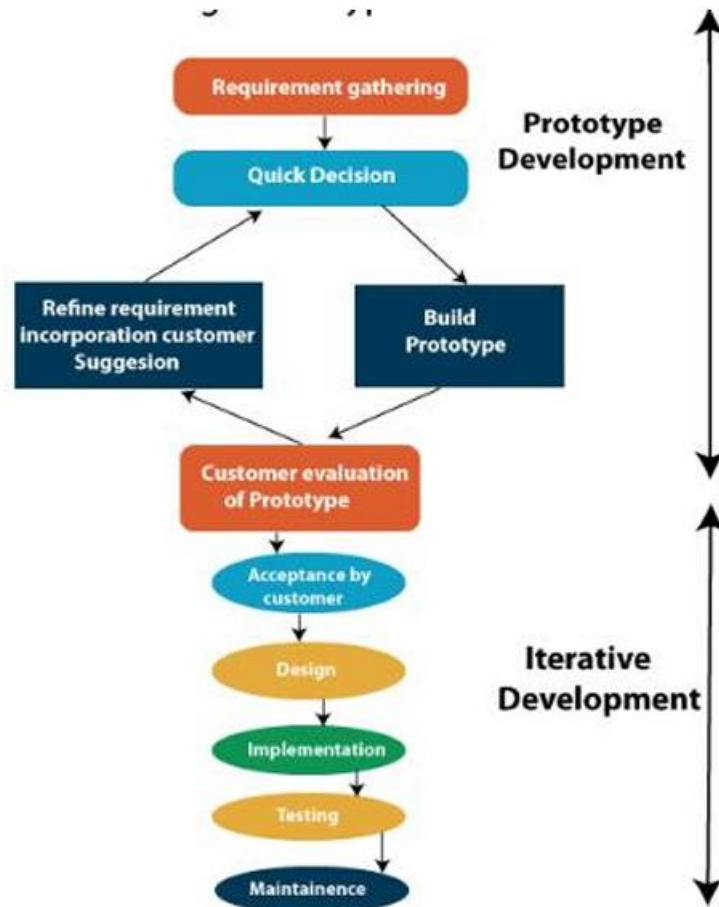


Fig.5.1 Prototype Model

The Software Development Life Cycle (SDLC) for Stayfinder web application typically consists of the following phases:

1. **Requirement Analysis:** Identified key functionalities like user authentication, listing module, map integration, reviews, and ratings.
2. **Design:** Created diagrams such as Use Case, ER, and Data Flow to outline the structure and flow.
3. **Implementation:** Developed the application using technologies like HTML5, CSS3, JavaScript, Node.js, EJS, and MongoDB.
4. **Testing:** Tested the application for functional accuracy and database performance.
5. **Deployment:** Made the application ready for deployment.
6. **Maintenance:** Ensured regular updates and fixes for optimal performance.

5.2 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high level functionality of a system and also tells how the user handles a system.

Purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

The Use Case Diagram outlines the interactions between users and the Stayfinder system. The primary actors are:

- **Guest User:** Can browse listings, view reviews, and sign up.
- **Registered User:** Can log in, create listings, leave reviews, and view maps.

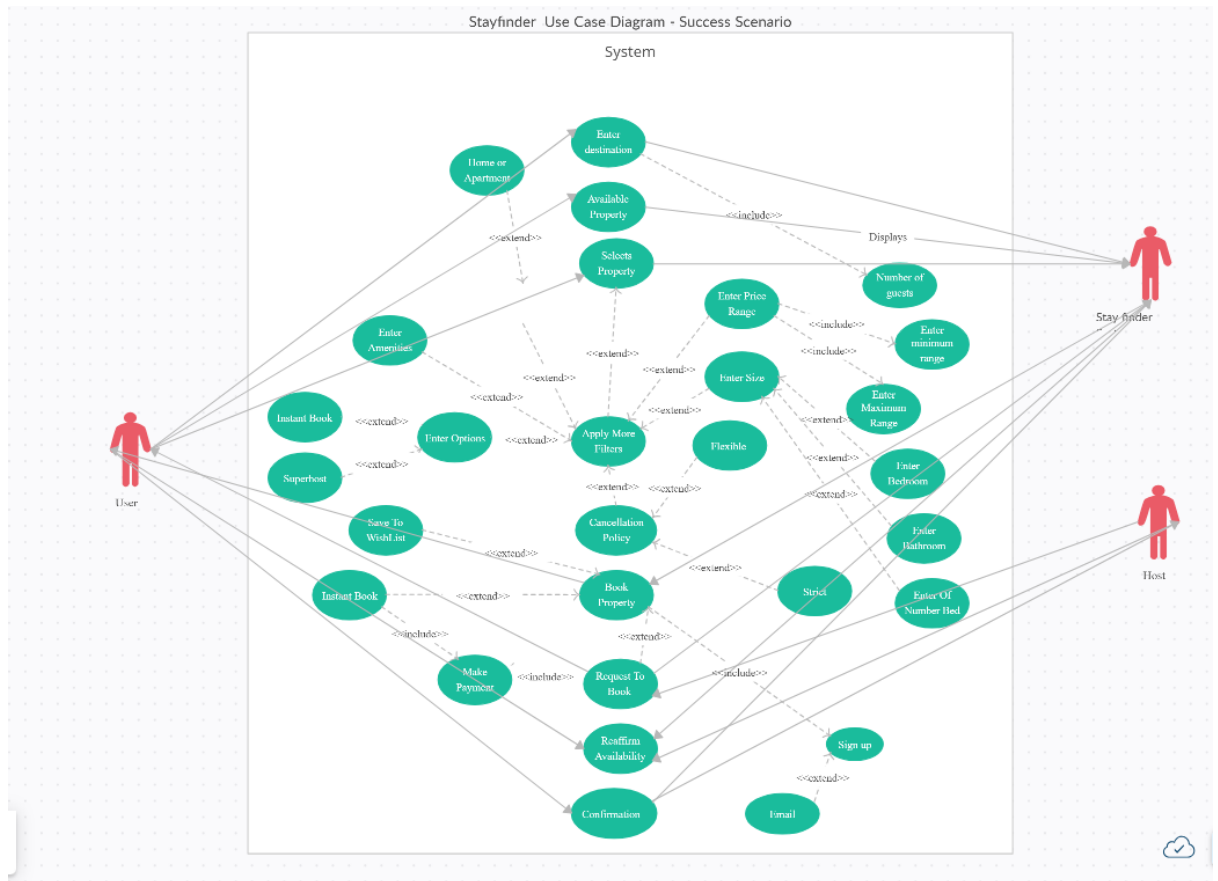


Fig.5.2 Use Case Diagram

5.3 ER (Entity Relationship) Diagram

1- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

2- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

3- In ER modeling, the database structure is portrayed as a diagram called an entity relationship diagram.



Fig.5.3. ER Diagram

5.4 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

Level 0 DFD

```
graph TD
    User -->|Input Data| System[Stayfinder System]
    System -->|Output Data| User
```

Level 1 DFD

```
graph TD
    A[User] -->|Signup/Login| B[User Module]
    B -->|Authenticated User| C[Listing Module]
    C -->|Listing Details| D[Review Module]
    C -->|Listing Location| E[Map Integration]
```

Level 2 DFD

```
graph TD
    User -->|Login Request| Auth[Authentication Service]
    Auth -->|Verify Credentials| Database[MongoDB]
    Database -->|Send User Data| Auth
    Auth -->|Success/Failure| User
    User -->|Request Listings| ListingService[Listing Module]
    ListingService -->|Fetch Listings| Database
    Database -->|Return Listings| ListingService
    ListingService -->|Send Listings| User
    User -->|Submit Review| ReviewService[Review Module]
    ReviewService -->|Save Review| Database
    User -->|Request Map| MapService[Map Integration]
    MapService -->|Fetch Coordinates| Database
    Database -->|Return Coordinates| MapService
    MapService -->|Show Map| User
```

Fig.5.4 Data Flow Diagram

5.5 Flow Chart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.



Fig.5.5 Flow Chart

Chapter 6

TESTING

6.1 Introduction

Testing is an essential phase in the software development lifecycle, ensuring that the application meets user expectations, functions correctly, and is free from critical bugs or vulnerabilities. For the *Stayfinder* project, a thorough testing strategy was crucial to guarantee the reliability, security, and usability of the platform. This chapter covers the comprehensive testing strategy adopted for *Stayfinder*, the types of testing conducted, and detailed test cases for the project's core modules: the signup and login module, listing module, review and rating module, and map integration.

Testing was conducted using a combination of manual and automated approaches, ensuring both the functional and non-functional aspects of the project were tested to meet industry standards. This chapter will include a detailed description of:

- The overall testing strategy and methodologies used.
- Different types of testing conducted.
- Specific test cases for each module.
- Results and analysis of the testing process.

6.2 Testing Strategy and Methodologies

The testing strategy for *Stayfinder* aimed to ensure that the platform's features functioned correctly, securely, and efficiently under various scenarios. The following methodologies were applied:

1. Manual Testing:

- **Purpose:** To simulate real user interactions and identify issues in usability, interface, and behavior.
- **Approach:** Testers followed predefined scenarios and input various data to assess the system's response.
- **Benefits:** Enabled detection of usability issues that automated testing might overlook.

2. Automated Testing:

- **Purpose:** To quickly run repetitive tests and validate that updates or new features do not break existing functionalities.
- **Tools Used:**
 - **Jest:** Utilized for backend unit tests to validate the server-side logic.
 - **Supertest:** For API endpoint testing to verify server responses and API behaviors.
 - **Cypress:** Used for end-to-end testing, simulating real user interactions to ensure the entire workflow from login to property search functions as expected.
- **Benefits:** Enhanced test coverage and speed, ensuring consistent testing with every code update.

3. Types of Testing Conducted:

- **Unit Testing:** Verified the functionality of individual functions, methods, and classes.
- **Integration Testing:** Checked the interaction between different modules, such as how the signup and login module interacted with the database.
- **System Testing:** Assessed the entire system to ensure that all components worked together seamlessly.
- **Acceptance Testing:** Performed to verify that the application met the project requirements and expectations.
- **Regression Testing:** Conducted after code updates to ensure that changes did not disrupt existing features.
- **Performance Testing:** Evaluated the platform's responsiveness under different load conditions.
- **Security Testing:** Ensured that authentication, data handling, and storage mechanisms were secure from vulnerabilities such as SQL injection and cross-site scripting (XSS).

4. Testing Environment:

- **Development Environment:** Local setup in Visual Studio Code, using tools like Postman for API testing.
- **Staging Environment:** A pre-production environment that mirrored the live server to test real-world scenarios.
- **Production Environment:** Testing on the live server to ensure that everything functions correctly after deployment.

6.3 Types of Testing Conducted

Functional Testing:

Objective: To validate that the *Stayfinder* application behaves as expected when users interact with the system.

Approach: Test cases were created to check each functionality, including user registration, property searches, booking process, and review submission.

Example Test Cases:

User Registration: Verified that a user could sign up using valid credentials and receive a confirmation message.

Search and Filtering: Confirmed that users could search for properties using various filters and criteria.

Review and Ratings: Ensured that reviews could be submitted, displayed, and rated properly.

Non-Functional Testing:

Performance Testing: Tools like Apache JMeter were used to simulate simultaneous users and measure the application's response times.

Security Testing: Implemented using automated scanners and manual checks to find vulnerabilities, including XSS and SQL injection. Security measures such as JWT token validation and password encryption were tested.

Usability Testing: Conducted to ensure that the user interface was easy to navigate and the overall experience was intuitive.

Usability and User Acceptance Testing (UAT):

Objective: To confirm that the application meets user expectations and is ready for deployment.

Method: Real users tested the application in a controlled environment and provided feedback.

Example:

Users were asked to complete tasks such as signing up, logging in, searching for a property, and submitting a review.

Feedback was collected, and necessary adjustments were made to improve usability.

Compatibility Testing:

Objective: To ensure that the application functions properly across different browsers and devices.

Method: Tests were run on browsers like Chrome, Firefox, and Safari, as well as on mobile devices.

Tools Used: BrowserStack was used for cross-browser testing to check the application's responsiveness and layout compatibility.

6.4 Detailed Test Cases for Each Module

1. Signup and Login Module Test Cases

Test Case 1: User Registration

- **Objective:** To verify that a new user can register successfully.
- **Steps:**
 1. Navigate to the signup page.
 2. Enter valid user details (e.g., name, email, password).
 3. Submit the registration form.
- **Expected Result:** The user is registered and redirected to the login page or receives a confirmation message.
- **Pass/Fail Criteria:** Pass if the user data is stored in MongoDB and a success message is shown.

Test Case 2: Login with Valid Credentials

- **Objective:** To ensure users can log in with correct credentials.
- **Steps:**
 1. Enter registered email and password.
 2. Click the "Login" button.
- **Expected Result:** The user is authenticated and redirected to the homepage.

Test Case 3: Login with Invalid Credentials

- **Objective:** To check the behavior when invalid login details are provided.
- **Steps:**
 1. Enter an unregistered email and/or incorrect password.
 2. Click "Login."
- **Expected Result:** An error message should be displayed, indicating invalid credentials.
- **Pass/Fail Criteria:** Pass if an error message is shown and the user is not logged in.

2. Listing Module Test Cases

Test Case 1: Adding a Property Listing

- **Objective:** To verify that property owners can successfully add a new listing.
- **Steps:**
 1. Log in as an owner and navigate to the "Add Listing" page.
 2. Enter required details (e.g., title, price, location, description, images).
 3. Submit the listing form.
- **Expected Result:** The new listing is added to the database and displayed in the listings section.
- **Pass/Fail Criteria:** Pass if the listing appears in the database and is visible on the listings page.

Test Case 2: Searching for Properties

- **Objective:** To check that users can search for properties based on filters.
- **Steps:**
 1. Enter search criteria (e.g., location, price range).
 2. Click "Search."
- **Expected Result:** The search results should match the criteria entered.
- **Pass/Fail Criteria:** Pass if the properties displayed meet the filter criteria.

3. Review and Rating Module Test Cases

Test Case 1: Submitting a Review

- **Objective:** To verify that users can submit reviews and ratings for a property.
- **Steps:**
 1. Navigate to a property page.
 2. Enter a review and rating.
 3. Submit the review.
- **Expected Result:** The review is added to the database and displayed on the property page.
- **Pass/Fail Criteria:** Pass if the review appears with the correct content and rating.

Test Case 2: Viewing Reviews for a Property

- **Objective:** To check that users can view all reviews for a property.
- **Steps:**
 1. Navigate to the property page.
 2. Scroll down to the review section.
- **Expected Result:** All submitted reviews and ratings are displayed.
- **Pass/Fail Criteria:** Pass if all reviews are correctly shown.

5. Map Integration Test Cases

Test Case 1: Displaying Property Locations on the Map

- **Objective:** To ensure that property locations are accurately displayed on the map.
- **Steps:**
 1. Open the map view.
 2. Confirm that property markers appear at their designated locations.
- **Expected Result:** The properties should be represented correctly on the map.
- **Pass/Fail Criteria:** Pass if all locations match the database coordinates.

Test Case 2: Zoom and Pan Functionality

- **Objective:** To verify that users can zoom in and out and pan the map seamlessly.
- **Steps:**
 1. Interact with the map by zooming and panning.
- **Expected Result:** The map should respond to user actions without lag or error.
- **Pass/Fail Criteria:** Pass if the map functionality works smoothly without glitches.

6.5 Results and Analysis

Testing Outcomes:

- The manual testing phase revealed usability issues such as form validation feedback, which were fixed to enhance user experience.
- Automated testing showed that API endpoints responded correctly under various scenarios, confirming the robustness of server-side logic.
- Performance tests showed that the platform could handle up to 500 concurrent users without a significant drop in performance.

Bug and Issue Tracking:

- All issues identified during testing were documented and assigned to the development team for resolution.
- Bugs were prioritized based on their severity, with critical bugs addressed first.

Conclusion: Testing was a crucial step in ensuring that *Stayfinder* functioned as intended. Each module was thoroughly tested, and all major issues were addressed. The comprehensive approach to testing ensured that the platform was reliable, secure, and met user expectations before deployment.

Chapter 7

PROJECT SCREENSHORTS

7.1 Home Page

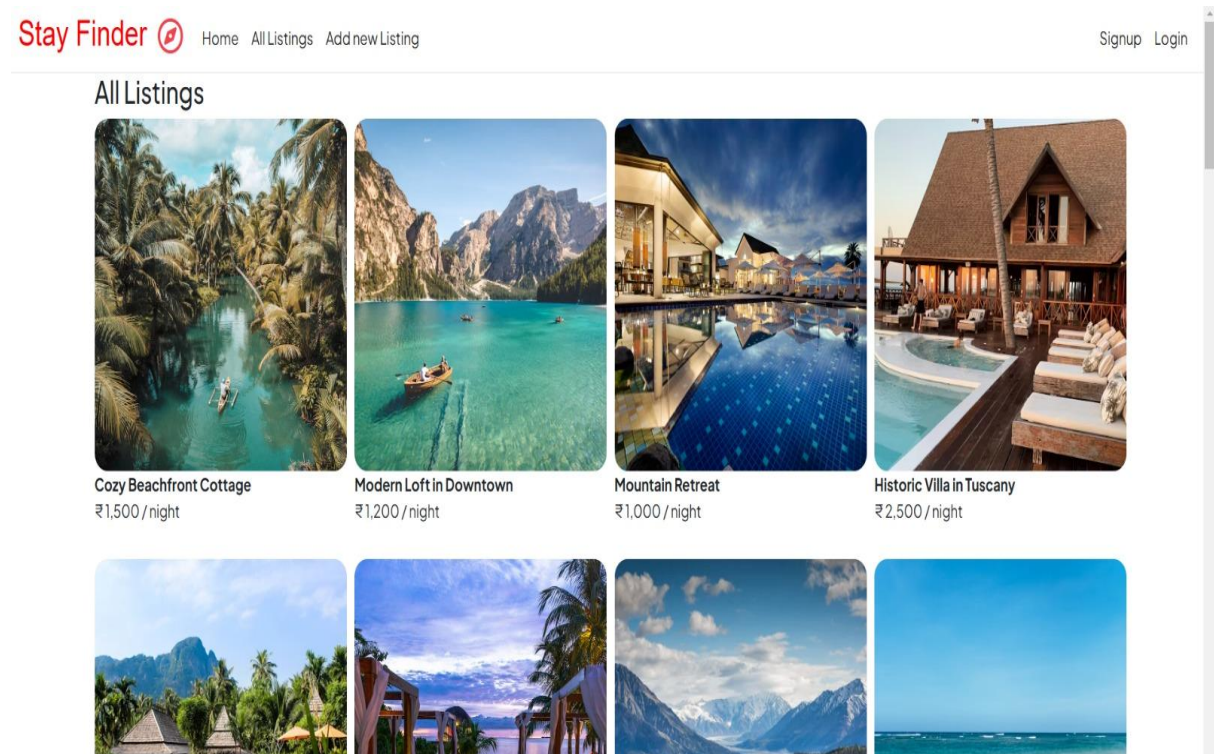



Fig.7.1. Home Page

7.2 Sign in

Stay Finder  [Home](#) [All Listings](#) [Add new Listing](#) [Signup](#) [Login](#)

Signup on Stay Finder

Username

Email

Password

[SignUp](#)


[in](#) [f](#) [@](#)

©Stay Finder Private Limited

[PrivacyTerms](#)

Fig.7.2 . Sign in

7.3 Login

Stay Finder  [Home](#) [All Listings](#) [Add new Listing](#) [Signup](#) [Login](#)

Login

Username

Password

[Login](#)

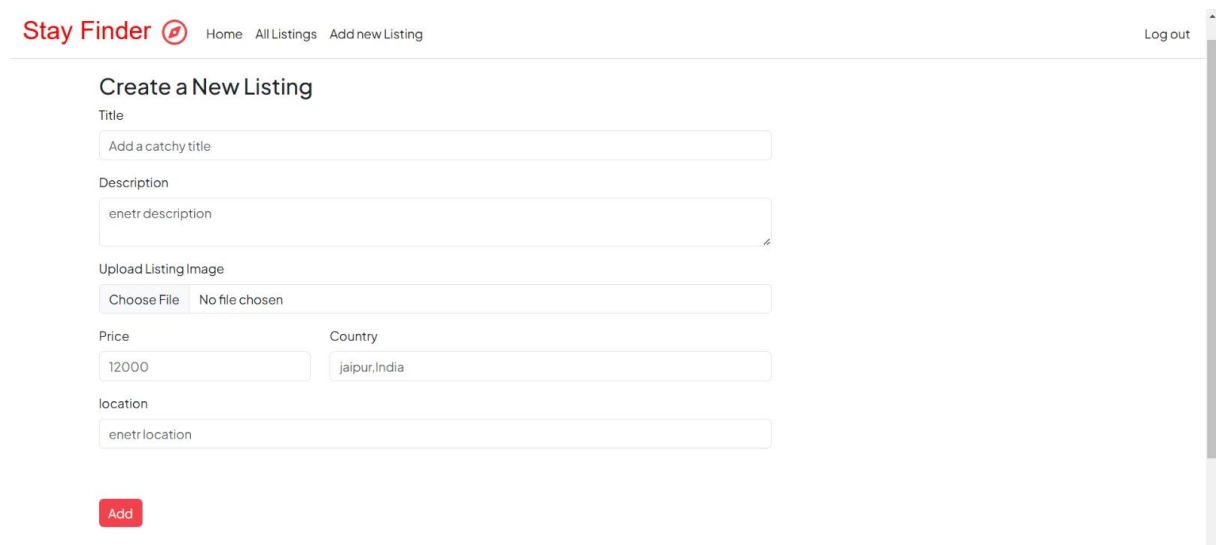
[in](#) [f](#) [@](#)

©Stay Finder Private Limited

[PrivacyTerms](#)

Fig.7.3. Login

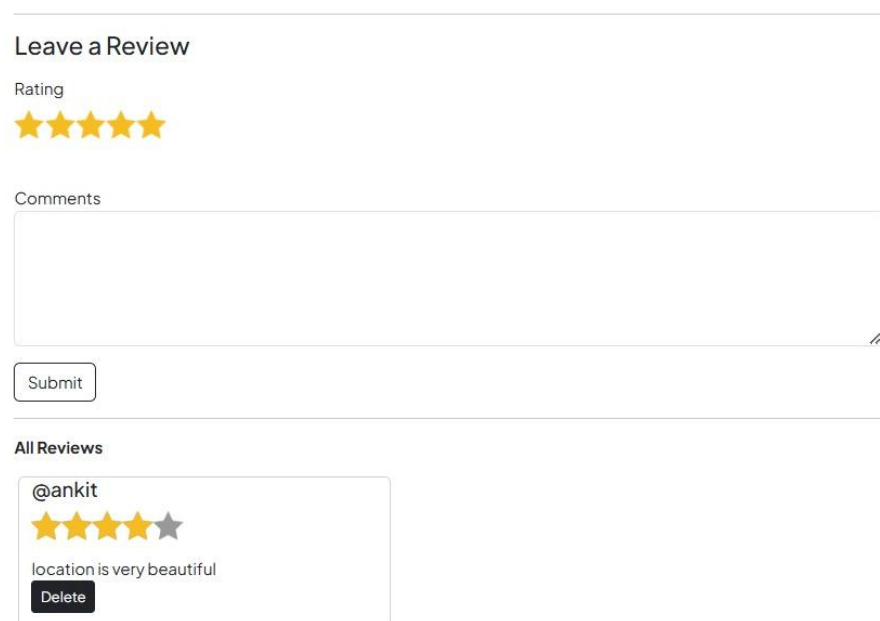
7.4 Create New Listings



The screenshot shows the 'Create a New Listing' form in the Stay Finder application. The form is located on the left side of the page, with a navigation bar at the top containing the 'Stay Finder' logo, 'Home', 'All Listings', 'Add new Listing', and a 'Log out' link. The form itself has a title 'Create a New Listing' and several input fields: 'Title' (placeholder: 'Add a catchy title'), 'Description' (placeholder: 'enetr description'), 'Upload Listing Image' (with a 'Choose File' button and 'No file chosen' text), 'Price' (placeholder: '12000'), 'Country' (placeholder: 'jaipur,India'), 'location' (placeholder: 'enetr location'), and a red 'Add' button at the bottom.

Fig.7.4. Create New Listing

7.5 Review and Ratings



The screenshot shows the 'Leave a Review' form and the 'All Reviews' section in the Stay Finder application. The 'Leave a Review' form is located on the left side of the page, with a title 'Leave a Review' and several input fields: 'Rating' (with five yellow stars), 'Comments' (placeholder: 'enetr location'), and a 'Submit' button. The 'All Reviews' section is located on the right side of the page, showing a list of reviews. The first review is from '@ankit' with a rating of four yellow stars and a comment 'location is very beautiful'. There is a 'Delete' button next to the review.

Fig.7.5. Review and Ratings

7.6 Map Integration

Where you'll be



©Stay Finder Private Limited

[PrivacyTerms](#)

Fig.7.7. Map Integration

Chapter 8

MAINTENANCE AND FUTURE SCOPE

8.1 Maintenance Objectives

Maintenance is a crucial phase in the software lifecycle that ensures the longevity, reliability, and performance of an application after its initial deployment. For the *Stayfinder* project, maintenance aims to:

- **Ensure System Reliability:** Regular updates and bug fixes are necessary to ensure that the platform remains stable and performs optimally.
- **Enhance Security:** Implementing patches and security updates to protect user data and safeguard against potential vulnerabilities.
- **Improve Performance:** Continuously monitoring and optimizing code and database queries to ensure fast response times and seamless user experiences.
- **Add Features and Functionality:** Iterative improvements based on user feedback to make *Stayfinder* more feature-rich and user-friendly.
- **Compliance with Regulations:** Keeping the platform up to date with data protection laws and web standards to maintain compliance.
- **Bug Management:** Timely resolution of reported issues to maintain a high level of user satisfaction.

The goal of maintenance is to extend the application's lifecycle and provide users with a consistent, valuable experience while adapting to changing technology and user needs.

8.2 Maintenance Strategies

For *Stayfinder*, the following maintenance strategies are adopted:

1. Proactive Maintenance:

- **Monitoring Tools:** Regular monitoring of server performance using tools like New Relic or AWS CloudWatch to detect anomalies and address them before they impact users.

- **Performance Benchmarking:** Implementing regular benchmarks using tools like Apache JMeter to assess the application's scalability and performance under heavy loads.
- **Scheduled Code Reviews:** Conducting code reviews at regular intervals to ensure coding standards are met, optimizing for readability, efficiency, and maintainability.

2. Corrective Maintenance:

- **Bug Tracking Systems:** Utilizing tools like Jira and GitHub Issues to log, track, and prioritize bugs and issues reported by users or identified by automated tests.
- **Patch Management:** Deploying patches for critical issues that affect the application's functionality, security, or performance.
- **Rollback Plan:** Having a plan in place for quick rollbacks in case a patch or update disrupts the system's normal operation.

3. Adaptive Maintenance:

- **Updating Dependencies:** Ensuring that all third-party libraries, such as Bootstrap, EJS, and Node.js modules, are kept up to date to leverage new features and security patches.
- **Compatibility Adjustments:** Modifying code to remain compatible with new web browsers and mobile operating systems.
- **Database Optimization:** Reassessing and optimizing database schemas and queries for efficiency as the dataset grows and evolves.

4. Perfective Maintenance:

- **User Feedback Integration:** Regularly collecting feedback through user surveys and analytics tools to identify areas for improvement.
- **Feature Refinements:** Enhancing the UI/UX based on user feedback to make navigation more intuitive and interactions smoother.
- **A/B Testing:** Implementing A/B tests to evaluate user response to new features and design changes, allowing data-driven decisions on improvements.

8.3 Maintenance Process

1. Issue Identification and Reporting:

- Users and testing teams can report issues through an issue-tracking platform like Jira.
- Automated monitoring tools provide real-time alerts for potential issues, such as high server load or response time delays.

2. Prioritization and Assignment:

- Issues are categorized into high, medium, or low priority based on their impact on the platform's functionality and user experience.
- High-priority issues are assigned immediate attention, while lower-priority tasks are scheduled for later.

3. Issue Resolution and Testing:

- Developers work on resolving reported bugs and implementing updates.
- Each update or fix is tested locally using unit tests and then pushed to a staging environment for integration and system testing.
- Quality assurance (QA) testers verify that the changes do not cause regressions or new issues.

4. Deployment and Monitoring:

- Once the update passes QA, it is deployed to the production environment.
- Continuous monitoring ensures that the new version operates as expected and does not introduce new issues.
- User feedback is collected post-deployment to gauge satisfaction and performance.

5. Documentation and Communication:

- All updates, changes, and patches are documented, including a summary of modifications, known issues, and the expected outcomes.
- Regular communication with users and stakeholders ensures transparency and builds trust.

6. Ongoing Review and Adaptation:

- Maintenance strategies are reviewed periodically to adapt to new technologies, user feedback, and performance metrics.
- Regular team meetings are held to discuss new maintenance protocols and ensure all team members are aligned.

8.4 Future Scope

The future development of *Stayfinder* includes adding new features, scaling the platform to support more users, and expanding to new technologies to stay competitive. Below are potential future enhancements:

1. Enhanced Search Capabilities:

- **Natural Language Processing (NLP):** Integrating NLP for smarter search queries, allowing users to find properties using conversational language.

- **AI-Powered Recommendations:** Using machine learning algorithms to provide personalized property recommendations based on user behavior and preferences.

2. Mobile Application Development:

- **iOS and Android Apps:** Developing native apps to offer a better user experience on mobile devices, using frameworks like React Native or Flutter.
- **Push Notifications:** Integrating push notifications for instant updates on property availability, booking confirmations, and new reviews.

3. Advanced Security Features:

- **Two-Factor Authentication (2FA):** Adding 2FA during the login process to enhance account security.
- **End-to-End Encryption:** Implementing end-to-end encryption for all communication and payment data to ensure data protection.

4. Integration with Third-Party Services:

- **Payment Gateways:** Adding support for more payment gateways like PayPal and Stripe to provide users with a wider range of payment options.
- **Social Media Login:** Allowing users to log in using social media accounts (e.g., Facebook, Google) for faster registration and login.

5. Real-Time Booking Updates:

- **Live Availability Check:** Integrating real-time property availability to avoid double bookings and ensure accurate data.
- **Instant Booking Confirmation:** Improving the user experience by providing immediate booking confirmations.

6. Multi-Language Support:

- **Localization:** Adding multi-language support to cater to a global audience and improve accessibility for non-English speaking users.
- **Region-Specific Customization:** Offering region-specific content, such as local attractions and cultural insights, to provide added value.

7. User-Centric Improvements:

- **User Dashboard Enhancements:** Expanding user dashboards with a history of past bookings, personalized suggestions, and enhanced user profile management.
- **Host and Guest Feedback System:** Creating a more sophisticated review and rating system for both hosts and guests to foster trust and transparency.

8. Performance Scaling:

- **Load Balancing:** Implementing load balancing to manage high traffic and ensure seamless user experience during peak times.
- **Microservices Architecture:** Transitioning to a microservices-based architecture to scale specific components independently and improve system resilience.

9. Data Analytics and Reporting:

- **Admin Dashboard:** Creating a robust admin dashboard with data analytics tools to help manage listings, bookings, and user activity.
- **User Insights:** Providing hosts with analytics about their listings' performance and feedback to optimize their offerings.

10. Machine Learning for Dynamic Pricing:

- **AI-Based Pricing:** Implementing an AI-driven pricing algorithm that adjusts property prices based on factors such as demand, seasonality, and local events.
- **Market Analysis:** Leveraging machine learning to analyze market trends and offer insights into pricing strategies for hosts.

11. Sustainability Features:

- **Eco-Friendly Filters:** Adding filters for properties that are environmentally friendly, such as those using renewable energy or promoting sustainable practices.
- **Green Partnerships:** Partnering with eco-friendly service providers to offer users options for sustainable accommodations and activities.

12. Collaboration and Community Features:

- **Forum and Chat:** Adding a community forum and in-app chat features for users to communicate, share tips, and coordinate with hosts.
- **Event Management:** Introducing event organization capabilities where hosts can manage gatherings and activities at their properties.

Chapter 9

CONCLUSION

Bibliography

- Airbnb. (n.d.). Retrieved from <https://www.airbnb.com>
- Apna College. (n.d.). Web development tutorials. YouTube. Retrieved from <https://www.youtube.com/c/ApnaCollege>
- MDN Web Docs. (n.d.). HTML5, CSS3, and JavaScript documentation. Retrieved from <https://developer.mozilla.org>
- Bootstrap. (n.d.). Official documentation. Retrieved from <https://getbootstrap.com/docs>
- Node.js. (n.d.). Official documentation. Retrieved from <https://nodejs.org/en/docs>
- MongoDB. (n.d.). Official documentation. Retrieved from <https://www.mongodb.com/docs>
- Embedded JavaScript (EJS). (n.d.). Documentation. Retrieved from <https://ejs.co>
- Stack Overflow. (n.d.). Community discussions and solutions. Retrieved from <https://stackoverflow.com>

References

The development of the Stayfinder project was made possible by leveraging various resources and references. These sources provided valuable guidance, tutorials, and insights that significantly contributed to the successful implementation of the project's modules and functionalities. Below is the list of references used for the project:

1. Airbnb Website

- The Airbnb website served as the primary inspiration for the Stayfinder project. Its design, user interface, and functionality provided a comprehensive understanding of how to create a platform for property listing and user interaction.

2. Apna College YouTube Channel

- Videos from the Apna College YouTube channel, particularly those focusing on web development, JavaScript, and Node.js, were immensely helpful in implementing the project's backend and frontend functionalities.

3. MDN Web Docs (Mozilla Developer Network)

- MDN Web Docs were extensively referred to for understanding HTML5, CSS3, and JavaScript standards and best practices. These resources helped ensure proper coding practices and the use of modern web technologies.

4. Bootstrap Documentation

- The official Bootstrap documentation was used for incorporating responsive design elements and pre-built UI components, which enhanced the overall user experience of the Stayfinder project.

5. Node.js and MongoDB Official Documentation

- Node.js and MongoDB documentation provided in-depth technical knowledge and examples for setting up the server, handling routes, and managing the database effectively.

6. EJS Documentation

- The Embedded JavaScript (EJS) documentation was referenced to implement templating functionalities, allowing dynamic content rendering on the Stayfinder platform.

7. YouTube Tutorials and Online Forums

- Various YouTube tutorials and forums like Stack Overflow were utilized to troubleshoot specific issues encountered during the development process and to gain additional insights into best practices for web application development.

MyLOFT E-Resources

