**BookWithMe**

**Online Rental Booking System**

#### By

**Prayag Desai**

**A project submitted**

**In**

**partial fulfillment of the requirements for the degree of**

**BACHELOR OF TECHNOLOGY**

**in Computer Engineering**

**Internal Guide**

*Prof.: Brijesh S. Bhatt.*

*Associate Professor*

Dept. of Comp. Engg.



**Faculty of Technology Department of Computer Engineering**

**Dharmsinh Desai University**

#### April 2019

CERTIFICATE

This is to certify that the project work titled

## BookWithMe

## Online Rental System

is the bonafide work of

Prayag Desai (Id no. 15CEUON030)

carried out in the partial fulfillment of the degree of Bachelor of Technology in Computer Engineering at Dharmsinh Desai University in the academic session

December 2018 to April 2019.

Brijesh S. Bhatt Dr. C. K. Bhensdadia

Assoc. Prof. Head,

Dept. of Computer Engg. Dept. of Computer Engg.



### Faculty of Technology Department of Computer Engineering

**Dharmsinh Desai University April 2019**

### Acknowledgements

I have taken efforts in this Project. However, it would not have been possible without the kind support and help of many individuals and organization. I would like to extend my sincere thanks to all of them.

I am highly indebted to Professor. **Brijesh S. Bhatt , Associate professor** and Professor. **Pandav Patel, Assistant Professor** of Department of Computer Engineering, Dharmsinh Desai institute of Technology Nadiad, for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express to my gratitude towards members of Dharamsinh Desai Institute of Technology, Nadiad for their kind co-operation and encouragement which helped me in completion of this project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| Chapter |  | Page |
|  | **Introduction**  Purpose of the document  Glossary  References | 5  5  5 |
|  | **About the System** | 6 |
|  | About BookWithMe  Purpose of the System  Scope of this System  Objective of the System  Tools and Technologies  System requirements | 6  10  12 |
|  | **Design and Analysis**  Workflow | 13  13 |
|  | **Implementation**  Code snippet with snapshots | 14  14 |
|  | **Implementation**  **Conclusions and Future Extesions** | 18 |

**Chapter 1**

**Introduction**

## 1.1. Purpose

The purpose of this document is to present a detailed description of BookWithMe the Online Rental Booking System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the External and the internal faculty for its approval.

## 1.2. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| System / BookWithMe | System / BookWithMe refers to the online web platform for booking rental at specifies place and time by its registered users, those properties hosted by the registered hosts. |
| Registered User | Person who is a member of the service provided by the System for booking rentals which are hosted by the registered hosts. |
| Database | Collection of all the information monitored by this system. |
| Admin | Person who overlooks and manages the registered users, registered hosts and non-registered users, host properties, user and host credentials and bookings. |
| Registered Hosts | Person who is a member of service provided by the System to declare his/her property for rent for specific period of time for registered users to stay. |
| Non Registered Users | Person who is not an active member of the services provided by the System either for booking rentals or hosting one. The non registered users can only view and look for available properties in particular location at particular time. Non registered users can become either registered users, registered hosts or both. |
| Review | A written recommendation made by the registered users who have stayed in the property, about the host property, host and his / her experience in the property. |
| Reviewer | A person that has stayed in the property hosted by registered hosts. |
| Software Requirements Specification (SRS) | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Stakeholder | Any person with an interest in the project who is not a developer. |

## 1.3. References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

**Chapter 2**

About the System

* 1. **About BookWithMe**

BookWithMe is an online platform for renting properties. BookWithMe helps homeowners to rent their spare rooms or entire property for about just days, weeks or months. ***BookWithMe*** will benefit homeowners to generate passive income on the their property which they no longer use. ***BookWithMe*** allows ***Non Registered Users*** to explore properties which are hosts by the ***Registered Hosts.*** The ***Registered Users*** can book properties for pre-specified period of time on location provided where ***BookWithMe*** has properties registered by ***Registered Hosts***. ***BookWithMe*** benefits the users as it is cheap compared to booking a motel and the service is excellent.

## Purpose of the System

The purpose of this System is to enable users to book properties which are made available by hosts through leveraging services provided by the System.

## Scope of the System

This software system will be an Online rental System. This system will be designed to minimize the renter’s tariff per rental booked as compared to the booking rates of motels and to maximize the earnings of the property owner who allow Registered Users to book their property for some time. The system does this by automating the management of property, bookings and credentials of hosts, users and properties which would otherwise have to be performed manually. The ***registered users*** are allowed to book the property made available by the ***registered hosts***, manage bookings (viewing booking details, deleting booking and managing bookings).The ***registered users*** can also manage their credentials, view property booked by them in the near past, make new bookings and view the specifications of the property booked by them or the property they intend to book. ***Registered users*** can also filter the property they search on the basis of hoe good a property is rated by others (filter properties by popularity), filter properties by certain class of property provided by the hosts (BookWithMe plus, entire home, boutique rooms and Unique homes.) and also filter property by the ascending order on the rates of rental per night. BookWithMe also implements a Review system which stores reviews and ratings given by the ***registered users*** to the property and the service of the ***registered host*** who host the property. This reviews made by the ***registered users*** are never deleted by the user and hosts. The review system is effective for honest feedback by the registered users. ***Registered hosts*** can make their property available for pre-specified period of time for the ***registered users*** to rent. ***Registered hosts*** can host property for booking, manage credentials, manage properties, extend/ delete bookings and make their properties unavailable for the renters. ***Non registered*** users can become registered users and registered hosts on successfully registering themselves for the same. ***Non registered users*** can only explore the properties available to rent but cannot book one. ***BookWithMe*** is a paid service. There is a one time fee for becoming registered users and registered hosts. The registered hosts have to pay pre-specified amount to the admin on every booking a registered user makes by using this platform. Also there is a booking cancellation fee for paid booking. This responsive application is made to run for both computers and mobile regarding any kind of operation system.

## Objective of the System

The objective of this System is to design and implement such a System which can run on multiple platform and browsers as the demand of online rental applications are certainly increasing. This software is made extensive user friendly and simple yet decent and powerful UI and feature rich which makes it very effective to use.

## Tools and Technologies used.

The System is implemented using Node.js and Express in the backend, html, CSS, JavaScript, jQuery and Bootstrap. No-SQL database MongoDB is database used by the System.

* + 1. **Why Node.js as backend?**

There are tons of alternatives to Node.js when building software but here are some reasons to use Node.js as a backend language.

* **JavaScript is probably the most popular programming language in the world**. It is the only language that can run inside a Web Browser, which gives it a unique point of advantage over any other language. With the rise of React Native, Electron and similar solutions, it’s also the only language which you can use to create web apps, frontend and backend, mobile apps, desktop apps. It’s pretty universal, and once you master the JavScript language in one area, your skills can be easily ported into another area and you just need to learn the environment differences (for example, how to use React Native rather than Electron, or the DOM).
* **Node.js is event driven and single threaded** What does this mean? In short, it can handle heavy load very well. Every single Node.js program can manage a lot of concurrent connections, and still be very fast because of its non-blocking I/O nature. This means that you have all the tools in place to build a very performant system, out of the box.
* **Node.js is hugely popular** This has a few side effects. First, you’ll find the solution to any kind of problem discussed online. **The community is huge** and very helpful. It’s easy to get started, and there are many resources to learn from. Second, there is a library for everything. The npm repository has a huge set of ready-to-use libraries, available with a simple npm install command. Thanks to this popularity, a vicious cycle is set up: your favorite API might only offer an official Node.js library, and other languages are not even considered, left for unofficial packages (with varying level of quality).
* **You can easily deploy Node.js apps** Some programming environments require a dedicated setup, even for simple projects, because no one has built an ecosystem around them.

* + 1. **Why Express with Node.js?**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

* Allows to set up middleware to respond to HTTP Requests.
* Defines a routing table which is used to perform different actions based on HTTP Method and URL.
* Allows to dynamically render HTML Pages based on passing arguments to templates.
  + 1. **Why Bootstrap in front-end?**
* **Saves time and is easy to use** Using Bootstrap, you can save a lot of time. You don’t have to spend time writing code, you can just use the Bootstrap predefined design templates and classes and put it exactly where they fit. That’s why it’s simple to use Bootstrap. If you have the basic working knowledge of HTML and CSS, you can start development with Bootstrap.
* **Customizable** An advantage of Bootstrap is that it offers many ways to be customized so that you can make it your own. You can pick and choose what is needed and toss what is not. Bootstrap can be adjusted exactly as you want it and as your project requires. This is simply accomplished using the Bootstrap customize page.
* **Great grid system** Creating page layouts needs a good grid. Bootstrap has this benefit: one of the best responsive, mobile grid system. It’s really easy to use and if you need to work through columns, then you’re in the right place using Bootstrap. Very handy when you want to hide some content based on screen size. Adding a class such as .visible-desktop to an element will make it visible only for desktop users. There are similar classes for tablets and phones
* **Consistency**
* **Responsiveness** The need to have a responsive website is very important. Creating mobile-ready websites is a breeze with Bootstrap thanks to the fluid grid layout that dynamically adjusts to the proper screen resolution. If you shift from a laptop to an iPad, you won’t have to worry over your work. Bootstrap adapts to the change in platforms.
* **Compatibility** Bootstrap is compatible with all modern browsers and Internet Explorer versions. Bootstrap is equipped with elements that are being considered the future of design itself. For example, both HTML5 and CSS3 are things that are going to be big in the future. Plugins like HTML5Shiv and Respond.js come as part of Bootstrap’s default template. These help in porting HTML5 elements into older non-HTML5 browsers.
  + 1. **Why MongoDB as Database?**

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

* MongoDB provides aggregation feature to use it in an efficient manner. For batch processing of data and aggregation operations, MapReduce can be used. MapReduce is nothing but an associated implementation for processing and generating big data sets with the parallel, distributed algorithm on a cluster.
* **MongoDB uses BSON format** One of the key features of MongoDB is that it uses BSON format. BSON is a JSON-like storage format. BSON stands for Binary JSON which is a binary-encoded serialization of JSON-like documents that MongoDB uses when storing documents in collections. It adds support for data types like Date and binary that aren’t supported in JSON
* **MongoDB Ad hoc queries** MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. MongoDB is able to support ad hoc queries by indexing BSON documents and using a unique query language
* **MongoDB is Schema – Less** MongoDB is a schema-less database (written in C++) because of which is much more flexible than traditional database tables. The benefit is the lack of setup and the reduced friction with OOP. So, in order to save an object, you just have to serialize it to JSON and send it to MongoDB. There is no need for type mapping which removes an additional burden.
* **MongoDB Indexing** Indexes are created to improve the performance of searches. The good thing is that any field in a MongoDB document can be indexed with primary and secondary indices. It enables the database engine to efficiently resolve queries which make it one of the best key features of MongoDB.
  + 1. **Why EJS as templating engine?**

EJS is the most popular templating engine for Node.js Express apps. Other popular options include PUG and handlebars.

* Use plain JavaScript We love JavaScript. It's a totally friendly language. All templating languages grow to be Turing-complete. Just cut out the middle-man, and use JS!
* Fast development time Don't waste time and attention figuring out arcane new syntax because 'elegance' — or how to preprocess your data so it will actually render right.
* Simple syntax JavaScript code in simple, straightforward scriptlet tags. Just write JavaScript that emits the HTML you want, and get the job done!
* Speedy execution We all know how fast V8 and the other JavaScript runtimes have gotten. EJS caches the intermediate JS functions for fast execution.
* Easy debugging It's easy to debug EJS errors: your errors are plain JavaScript exceptions, with template line-numbers included.
* Active development EJS has a large community of active users, and the library is under active development. We're happy to answer your questions or give you help.

## Tools used for developing this System.

This application is made by using VSCode editor. VSCode is a very powerful editor when it comes to working with multiple technologies. VSCode support multiple plugins which makes the development of front-end and back-end easier by intellisese. Nodemon is the server which is a development dependency used for testing the code and maintaining server while testing the app while production. MongoDB Compass is used for database modelling and visualize and explore data with ad-hoc queries. Debugging is carried out by the Chrome Debugger Tools of the Chrome Web Browser.

# **2.6 SYSTEM ANALYSIS**

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

## Requirement of the System

1. **User Requirements**

User requirements does not include many things, but most importantly user must be aware that system works properly with full availability, reliability, security and safety. The user responsibility are as follows:

Should know how to use the application and should adhere to the guidelines and prescribed standards.

1. **System Requirements**
2. **Functional Requirements**

**R1: Authentication for Registered Users**

**Input: Credentials**

**Description:** System gets the credentials of the registered users and according to the credentials of the user system gets the user information from the database.

**R1.1: System reads the username and password entered by the User**

**Input:** Correct credentials

**Output:** User is logged in.

**R1.1: System reads the username and password entered by the User**

**Input:** Incorrect credentials

**Output:** User is not logged in.

**R2: Authentication for Registered Hosts**

**Input: Credentials**

**Description:** System gets the credentials of the registered hosts and according to the credentials of the hosts system gets the host information from the database.

**R1.1: System reads the username and password entered by the Host**

**Input:** Correct credentials

**Output:** User is registered as registered Host. The credentials are stored in the Host database

**R1.1: System reads the username and password entered by the Host**

**Input:** Incorrect credentials

**Output:** User is not registered.

**R2: Registration for becoming Registered Hosts**

**Input:** Form credentials entered by non-registered users and/or registered users.

**Description:** System gets the credentials of the non-registered users and/or registered users. The system checks if there are Registered Hosts already present in the database with the entered credentials

**R1.1: If there is host with the same credentials entered**

**Input:** Form credentials entered by non-registered users and/or registered users.

**Output:** User is asked to enter the form again with new credentials.

**R1.1: If there is no host with the same credentials entered**

**Input:** Form credentials entered by non-registered users and/or registered users.

**Output:** User is registered as Registered Host

.

**R3: User can book and explore properties**

**Input:** Form data entered by the user which contain fields for check-in-date, check-out-date, accommodation per rental and the name of city and state.

**Description:** Users can book and explore properties in specified city, state and book properties on number of users accommodated in the rental

**R3.1 If there are no properties by user specification**

**Input:** Form data entered by the user which contain fields for check-in-date, check-out-date, accommodation per rental and the name of city and state.

**Output:** No properties

**R3.2 If there are properties demanded by the user**

**Input:** Form data entered by the user which contain fields for check-in-date, check-out-date, accommodation per rental and the name of city and state

**Output:** List of properties demanded by the user along with the property details

**R4: User can change their credentials**

**Input:** User’s updated credentials

**Description:** User can change their credentials and update their new credentials

**Output:** Conformation of updation of user credentials.

**R5. Registered users can delete booking**

**Input:** Request for deleting bookings

**Description:** User can delete booking made by them earlier. Users have to pay for the cancellation of the paid property and the cancellation of free property is not charger by the system

**R5.1 If the cancellation is paid**

**Input: User id, Payment id and cancellation scheme**

**Output:** The booking is deleted on payment of cancellation fee by the User.

**R5.2 If the cancellation is free**

**Input:** User id, Payment id and cancellation scheme

**Output:** The booking is deleted.

**R6. Registered users can view booking details.**

**Input:** Request for details on user’s bookings

**Description:** User can view booking details for the booked properties. Booking details consists of booking transaction id, property details which is booked by the user and when the property is booked by the user

**Output:** Booking Details.

**R7.** **Registered Users can view properties booked by them earlier**

**Input**: Request for viewing previously booked properties

**Description**: Registered Users can view properties booked by them earlier

**Output**: Properties booked by the user earlier. The properties has details on whether the property was successfully vacated by the user or the property was cancelled by the user

1. **Non Functional Requirements**

* **Scalability**

The system should able to scale up and scale down as per the traffic and need to optimize the operational cost.

* **Reliability and fault tolerance**

1. The system should up and running all the time with less to zero downtime.
2. The data should persist even after a catastrophic failure occurs.

* **Security**

User’s private data and system’s data should not be leaked or accessed by an unauthorized person.

* **Usability**

Be easy for users to understand and use the product.

* **Performance**

Be able to withstand the changing load and being fast and responsive all the time for users.

**Chapter 3**

Analysis

## 3.1 System Environment

Registered Users

Non-Registered users

Registered Users

Properties

Review and Ratings

Online Rental System

Registered Hosts

BookWithMe an Online Rental System has four active actors and one cooperating system.

The Author, Reader, or Reviewer accesses the Online Journal through the Internet. Any Author or Reviewer communication with the system is through email. The Editor accesses the entire system directly. There is a link to the (existing) Historical Society.

## 3.2 Use cases and Use case Diagram

## Use Case Diagram

## Figure-2. Use case diagram for BookWithMe

## 

This section outlines the use cases for each of the active actors separately. The actors for this system are Registered Hosts, Registered Users, Admin and Non registered users.

### 3.2.1 Actors

There are 4 actors interacting with this System Admin, Registered Users, Non-Registered User and Registered-Host

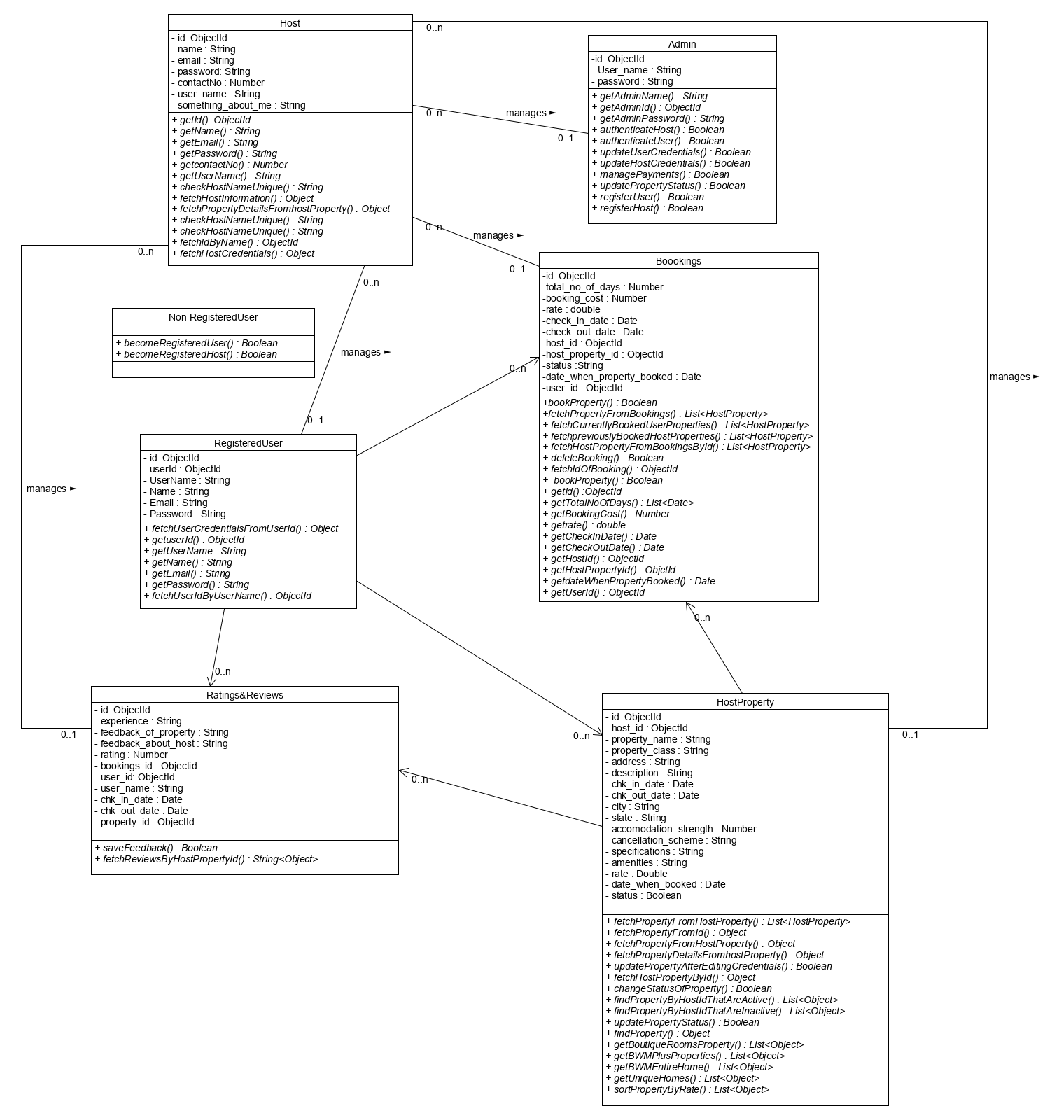
* + 1. Use cases
       1. Admin
          - Admin manages bookings, properties, review and ratings, payments, registered users, registered hosts, and their credentials.
       2. Registered Users
          - Registered Users can change their credentials.
          - Registered can book and explore properties by location and by location and tentative check-in and check-out dates and accommodation strength per rental
          - Registered Users can filter properties by property types and also filter property according to cheapest to costliest rate of rental.
          - Registered users can manage Bookings.
          - Registered users can view booking details.
          - Registered Users can delete the bookings. Registered users has to pay for the property that has paid cancellation and cancel property without payment for property that has free cancellation scheme
          - Registered users can view the properties that he/she has deleted or the properties that have been booked by the user previously.
          - Registered Users can also become Registered Host by signing up for Registered Host and rent his/her property.
          - Registered Users can make rating and give reviews for the properties that User has stated. Users cannot rate or review properties that the user has booked but not stayed still.
       3. Non-registered Users
          - Non registered users can view properties across various locations based on the check-in date, check-out dates and accommodation strength per rental.
          - Non registered users can become Registered Users upon registration.
          - Non registered users can become Registered Hosts upon registration.
       4. Registered Hosts
          - Registered Hosts can edit their credentials.
          - Registered Users can host their property for rent by Registered users for specific period of time by giving details about the property.
          - Registered hosts can delete the property they have hosted. Registered hosts are not allowed to update the property details like the address of the property, rate, cancellation scheme and the property class of such properties that has already booked by the Registered users but the Host can extend the date of availability of the property whereas the Registered hosts can change all the information about the property that has not booked by any Users.
          - Registered Hosts can make their property unavailable to book for registered users. The property can be made available to Users to book after adding and updating information about the property if the Host wish to make it available to the users to rent later.
          - Registered Host cannot make property unavailable which is already booked by the User. The host can extend dates for making the properties available.
          - Registered Host can view the number of bookings made in his property by the Registered Users. The Registered Hosts can then delete the bookings from his property once the user has vacated the property.
          - Registered Hosts can know what users has commented and rated and reviewed his property

**3.3 Data Flow Diagram**

A **data-flow diagram** (DFD) is a way of representing a flow of a data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops . Specific operations based on the data can be represented by a flowchart**.**

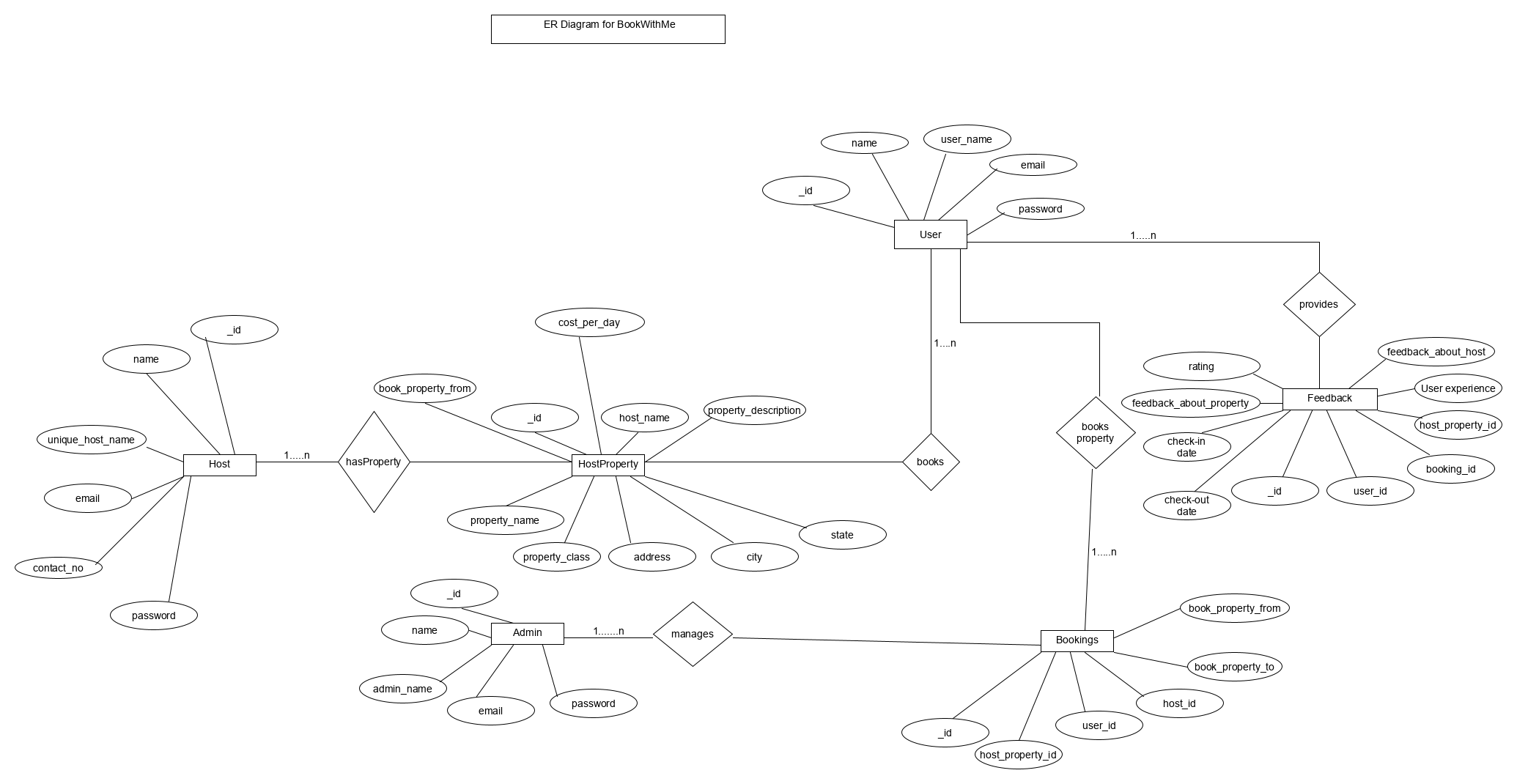
* 1. **Class Diagram**

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



**3.4 E-R Diagram**

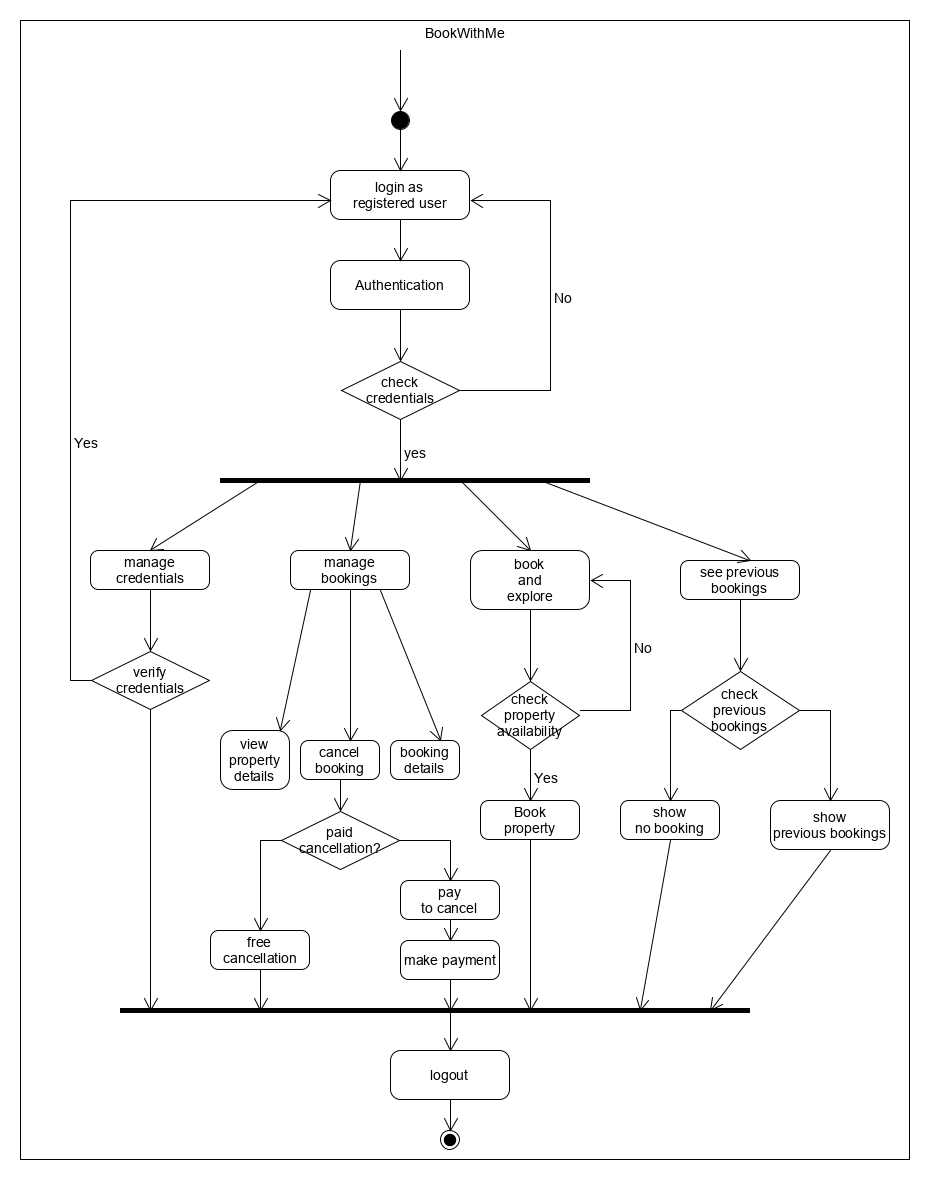
An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database.



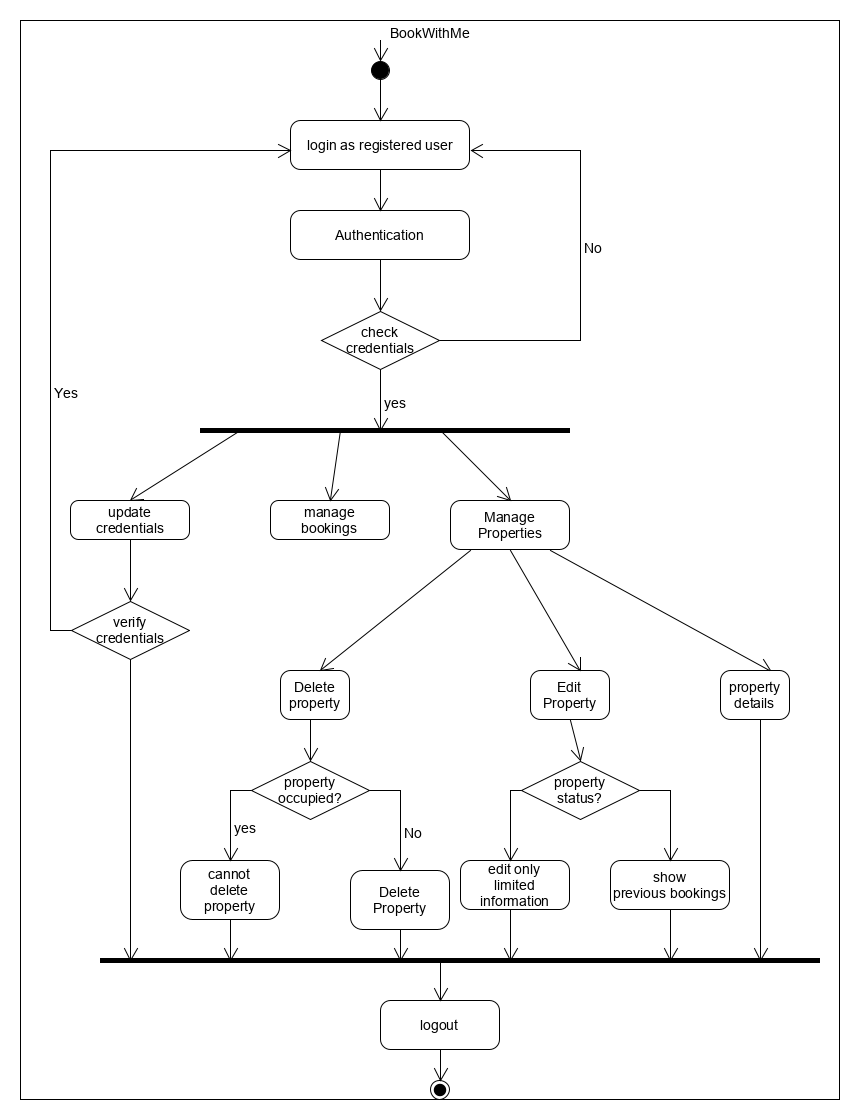
* 1. **Activity Diagram**

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

* + 1. **Activity Diagram for Registered Users**



## 3.5.2 Activity diagram for Registered Hosts



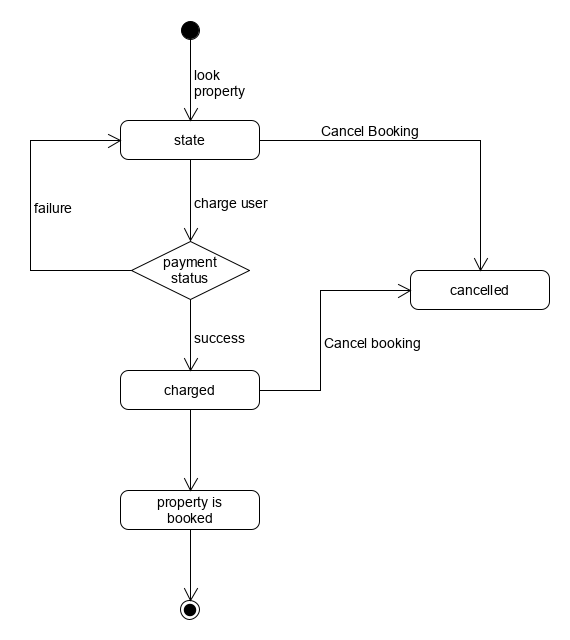
## Activity Diagram for Non-Registered users

## 

## State Diagram

A **state diagram** is used to represent the condition of the system or part of the system at finite instances of time. It’s a **behavioral** diagram and it represents the behavior using finite state transitions. It is used to state the events responsible for change in state (we do not show what processes cause those events).

## State diagram for booking property



## State diagram for cancelling booking

## 

## Sequence Diagram

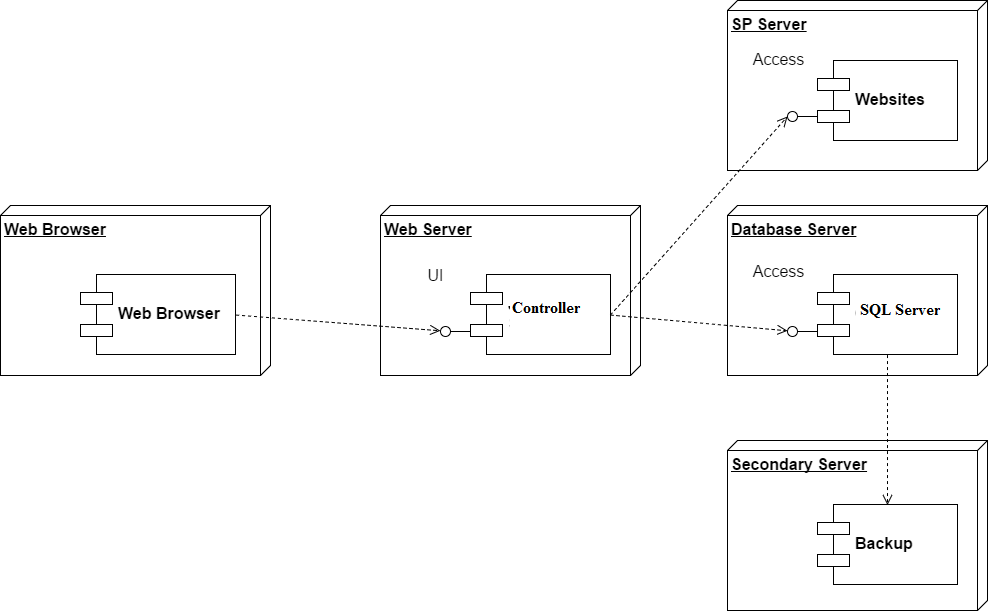
## Component Diagram

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

**C:\Users\vaibh\Downloads\state diagram\Component Diagram.png**

## Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

****

**CHAPTER 4**

**Design**

## 4.1 Database Design

The following tables are the part of the BookWithMe Database

1. User
2. Host
3. Admin
4. Feedback
5. Host Property
6. Booking

**4.1.1 Data Dictionary**

A data dictionary, or Meta data repository, as defined in the IBM. Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. The term may have one of several closely related meanings pertaining to databases and database management systems (DBMS):

* + - A document describing a database or collection of databases.
    - An integral component of a DBMS that is required to determine its structure.
    - A piece of middle ware that extends or supplants the native data dictionary of DBMS.

**Table 1. User table**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **D DATA TYPE** | **SPECIFICATION** | **DISCRIPTION** |
| \_id | ObjectId | Primary Key | User Unique Id. |
| Name | String |  | Name of the User |
| Username | String | - | Unique Username of the User |
| E-Mail | String | - | E-Mail id of the user |
| Password | String | - | Password for the User to log-in to the System. |

**Table 2. Host table**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **D DATA TYPE** | **SPECIFICATION** | **DISCRIPTION** |
| \_id | ObjectId | Primary Key | Host Unique Id. |
| Name | String |  | Name of the Host |
| Username | String | - | Unique Username of the Host |
| E-Mail | String | - | E-Mail id of the Host |
| Password | String | - | Password for the Host to log-in to the System. |
| Info | String | - | The info field stores the general introduction the Host gives to the Users which Users can see at the time of booking the property |

**Table 3. HostProperty table**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **D DATA TYPE** | **SPECIFICATION** | **DISCRIPTION** |
| \_id | Integer | Primary Key | Orders Unique Id. |
| HostId | ObjectId | Foreign Key | Id of the Host of the property. |
| PropertyName | String | - | Name of the property |
| PropertyClass | String | - | Describes the type of the property |
| Address | String | - | Address where property is located |
| Available-from | Date | - | Date from which the user can book this property |
| Available-to | Date | - | Date to which the user can book the property |
| State | String | - | Name of the State |
| City | String | - | Name of the City |
| Accomodation Strength | Number | - | The maximum number of guests allowed per rental property. |
| CancellationScheme | String | - | Specifies if the cancellation is Paid or Free cancellation of the property |
| Amenities | String | - | Specifies what amenities are provided on booking this rental property |
| Specifications | String | - | What is the most peculiar thing about the property |
| Rate | Float | - | Rate of the rental property per night. |
| Status | String |  | Shows if the property is open or not for the booking by the users |
| DateUpdated | Date |  | This shows when the property information was update by the Host of this property |

**Table 4. Booking table**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **D DATA TYPE** | **SPECIFICATION** | **DISCRIPTION** |
| \_id | ObjectId | Primary Key | Unique Booking Id. |
| TotalDays | Integer |  | Total no of days for which the booking is made |
| Amount | Float | - | Booking total amount |
| Check-in-Date | Date | - | Users Check-in-Date |
| Check-in-Date | Date | - | Users Check-out-Date |
| HostPropertyId | ObjectId | Secondary Key | Property Id of which booking is made |
| HostId | ObjectId | Secondary Key | Id of the Host to whom the property belong which is being booked |
| PropertyIsBooked | Boolean | - | Shows the status of the booking. Shows if this Booking is active or not |
| DateOfBooking | Date | - | Date when the booking is made. |
| UserId | ObjectId | - | Id of the User who made the booking. |

**Table 5. Feedback table**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **D DATA TYPE** | **SPECIFICATION** | **DISCRIPTION** |
| \_id | ObjectId | Primary Key | Unique feedback Id. |
| Experience | String |  | Overall experience about the host service and the Host property |
| PropertyFeedback | String | - | Feedback about the Property |
| HostFeedback | String | - | Feedback about the Host |
| BookingId | ObjectId | Secondary Key | Id of Booking to which the reviews and ratings are given |
| Rating | Integer | - | Overall ratings in the form of stars are given to the property |
| UserId | ObjectId | - | Id of the User who gave the ratings and review. |
| PropertyId | ObjectId |  | The id of the property to which the review and ratings are given |

## 4.2 Application Navigation

## 

The application is broken down into three Pieces. The Registered Users, Registered Hosts and The Non-Registered Users. Each one of then have different Navigations. All the navigations are handled by the Navigation Bars in the System. Others special navigations are handled by the link buttons.

#### 4.2.1 Navigation for Non-Registered Users

Non Registered Users can navigate to perform following actions

* User log-in
* User-Signup
* Host-Login
* Become Certified Host
* Help

**4.2.2 Navigation for Registered Users**

Registered Users can navigate to perform following actions

* Manage your Credentials
* Manage Bookings
* Book and Explore properties
* Show my previous Bookings
* User Dashboard

**4.2.3 Navigation for Registered Hosts**

Registered Hosts can navigate to perform following actions

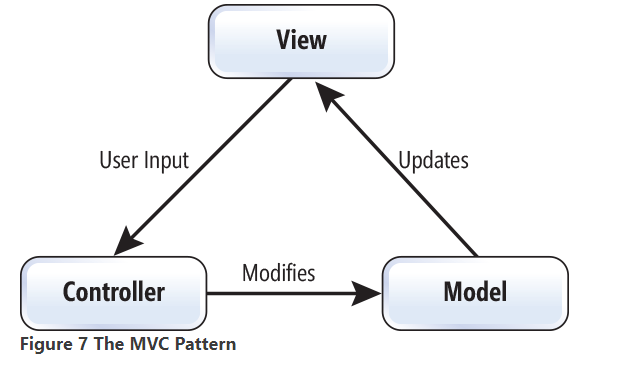
* Navigate to Host Dashboard
* Manage Credentials
* Create Rentals
* Manage Properties
* View Inactive properties

**4.3 Architectural Designing of the System**

This system is implemented by using MVC Architecture.

**Why MVC?**

Model–View–Controller (MVC) is an architecture that separates the representation of information from the user's interaction with it. It is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development



1. **Model**

Model code is written in some high-level programming language such as Java, Python, or C++. Usually the model code interfaces with a database using the Standard Query Language (SQL). In our system the model is implemented by Node.js

1. **View**

View represents the visualization of the data that model contains. The views in the System are EJS views.

1. **Controller**

The controller part is handled by Node.js at the back end. Node.js manages to control the business logic and is a server code that gets invoked whenever a user clicks on any link in the web pages or interacts enters data and submits a form.

**CHAPTER 5**

**Implementation**

# **5.1 PROJECT MANAGEMENT**

Project management is the practice of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria at the specified time. A project is a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or staffing) undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of such distinct production approaches requires the development of distinct technical skills and management strategies.

The primary challenge of project management is to achieve all of the project goals within the given constraints, this information is usually described in project documentation, created at the beginning of the development process. The primary constraints are scope, time, quality and budget. The secondary and most ambitious challenge is to optimize the allocation of necessary inputs and apply them to meet pre-defined objectives. The object of project management is to produce a complete project which complies with the client's objectives.

* **Life cycle used in project**

Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their users It advocates adaptive planning, evolutionary development, empirical knowledge, and continual improvement, and it encourages rapid and flexible response to change. Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their users It advocates adaptive planning, evolutionary development, empirical knowledge, and continual improvement, and it encourages rapid and flexible response to change.

The term agile was popularized, in this context, by the Manifesto for Agile Software Development. The values and principles espoused in this manifesto were derived from and underpin a broad range of software development frameworks, including Scrum and Kanban.

## 5.2 Feasibility study

At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

### Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

● Does the necessary technology exist to do what is suggested?

● Do the proposed equipment’s have the technical capacity to hold the data required to use the new system?

● Will the proposed system provide adequate response to inquiries, regardless of the number or location of customers?

● Can the system be upgraded if developed?

● Are there technical guarantees of accuracy, reliability, ease of access and data security?

The current system developed is technically feasible. It is GUI based user interface for online rental system. Thus, it provides an easy access to the client. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified.

### **Time Feasibility**

Time schedule plays a vital role in the project. If the project is not delivered on due time, then it can cause a project failure. Hence, time management should be taken care of by the project manager.

### **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

Is there sufficient support for professional service management system the management from the users?

Will the system be used and work properly if it is being developed and implemented?

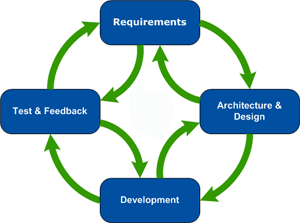
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

### **Implementation Feasibility**

Implementation feasibility is concerned with specifying external resources and software that will successfully satisfy the requirements. More importance is given to external resources and configuration of the system rather than the actual map of the hardware. Financial consideration is also considered at this stage.

## 5.3 Project planning



**Figure 1 Agile Cycle**

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. Project planning is often used to organize different areas of a project, including project plans, workloads and the management of teams and individuals.

### **Project development approach and justification**

* **System Analysis**

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyzes the scope of the project and plans the schedule and resources accordingly.

* **Software Design**

At this step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

* **Coding**

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

* **Testing**

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user’s end. Early discovery of errors and their remedy is the key to reliable software.

* **Integration**

Software may need to be integrated with the libraries, databases and other programs. This stage of SDLC is involved in the integration of software with outer world entities.

* **Implementation**

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

* **Maintenance**

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

In the development of this project, we will first check to see if our project is feasible functionally, technically and economically. Then we collect the requirements from the end users and analyze it. We also analyze similar systems to get an exact idea of how to create this system. Hence, we gather all the requirements which we need to develop our system. Then, after thoroughly understanding the needs of end user, we will develop the Graphical User Interface (GUI).

The GUI is viewed by the user and the user communicates with the system and hence, it should be appealing and attractive. After this comes the coding part, which involves handling databases and manages queries and forms etc. There are certain coding standards to be followed so that the flow of program is easily understood.

Testing will ensure that our system will work efficiently using all valid values and does not give errors. To test the system, we have to perform unit testing, module testing and the finally the system testing.

To maintain system up to date with the changes in the organization and ensuring it meets the goals of the organization by implementing changes to the system when necessary.

### **Milestones and Deliverables**

Milestones are identified in order to complete the entire project in the time duration. Milestones are identified for every module of this application developed.

**Table 1 Milestones and Deliverables**

|  |  |  |
| --- | --- | --- |
| **PHASE** | **DELIVERABLES** | **PURPOSE** |
| System Requirement and  Analysis | * Requirement Gathering and analysis. * Functional Specifications * Non Functional Specifications | It gives the exact understanding of the user’s requirements. |
| System Design | * E-R diagram * Data flow diagram * Use case diagram * Class diagram * Activity diagram | It gives the logical structure that describes the system. |
| Implementation  and Testing | * The output obtained for the required   Functionality after implementing and doing various types of testing | It gives the  required module |

### Roles AND RESPONSIBILITIES

**Table 2 Roles and Responsibilities**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Role** | | | | |
| **Analysis** | **Designing** | **Coding** | **Testing** | **Documentation** |
| Prayag Desai |  |  |  |  |  |

### **Project Scheduling chart**

**Table 3 Gantt Chart**

|  |  |
| --- | --- |
| **Software Life Cycle Phase** | **Duration** |
| **Analysis :** | **36 days** |
| 1. Requirement Gathering and learning Technology Stack | 28 days |
| 1. Feasibility Study | 2 days |
| 1. Planning | 4 days |
| 1. SRS | 2 days |
| **Designing :** | **30 days** |
| 1. E-R Diagram and Data Dictionary | 2 days |
| 1. Use Case Diagram | 2 days |
| 1. Class Diagram | 2 days |
| 1. Data Flow Diagram | 4 days |
| 1. Front-End validations and Changes | 15 days |
| 1. Form Designing | 5 days |
| **Implementation :** | **45 days** |
| 1. Back-end-Coding | 30 days |
| 2. Front-end-coding | 15 days |
| **Testing :** | **3 days** |
| 1. List Out Errors | 3 days |
| 1. Correction Of Errors | 4 days |
| **Documentation :** | **7 days** |

#### 5.4 Code Snippets

#### 

This System is Implemented using MVC Architectural pattern. The MVC modules are not hard coded in Node.js. Node.js Express app contains app.js file which contains the configurations on how to use middleware and all the third-party packages

In the ***app.js*** file

const express = require('express');

1. This is how third party packages are sed in NodeJS project.

const app = express();

1. To use the express in the app use a constant that remains the same in the application so that there are no problems in the naming conventions

app.set('view engine','ejs');

app.set('views','views');

1. This how templating engine is set in NodeJS express project

app.use(express.static(path.join(\_\_dirname, 'public')));

1. Set the static path in the express projects. This code creates a folder public in the project folder. Public folder is used to keep the css javascript and all the static files needed for the project

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(session({secret : 'prayagdesai'}));

1. The above code sets the body-parser which is used in Express to maintain cookies and sessions.

const adminRouter = require('./routers/admin');

1. This code is used to load the controller to the app.js. Every request made to the server is handled by the middleware in sequence. The middleware then executes the request and renders the page as requested

app.use(nonRegisteredUsersRouter);

app.use(adminRouter);

app.use(registeredHostsRouter);

app.use(registeredUsersRouter);

app.use(errorRouter);

1. These are the middlewares that handles the incoming request to the server.

The ***database.js file in the utility folder*** has the connection to the database. The database server will get connected once the back-end server gets connection.

The database will be created if there is no database or if there us a database, then the operation on the database will be performed.

Database.js

const mongodb = require('mongodb')

const mongoClient = mongodb.MongoClient; *// calling the mogodb client costructor*

let url = "mongodb://localhost:27017/demodbTest";

let \_db *// this underscore variable is used in this file only to fetch the type of database schema*

const mongoconnect = (callback) => { *// callback function passed on creating and connecting application to the database server*

mongoClient.connect(url,{useNewUrlParser:true}).

then(client =>{

console.log('Connected');

\_db = client.db()

callback(client);

})

.catch(err =>{

console.log('Error in connecting database');

*throw* err;

})

}

*// this function connects to the database \_db or else returns error on not finding database*

const getDb = () =>{

*if*(\_db)

{

*return* \_db;

}

*else*{

*throw* 'No database found';

}

}

exports.mongoconnect = mongoconnect;

exports.getDb = getDb;

Every incoming request is in the form of URL and the Router matches the URL and subsequently the Controller handles the URL

There are separate routers for Registered Hosts, Registered Users, Non-Registered Users and Admin

const express = require('express');

const router = express.Router();

const adminController = require('../controllers/admin\_c');

router.post('/registerhost',adminController.postbecomeHost);

router.post('/user-registration-successfull',adminController.postbecomeUser);

router.post('/user-auth',adminController.postAuthenticateUser);

router.post('/host-auth',adminController.postAuthenticateHost);

module.exports = router;

Routing is carried away by Router( ) of the Express Library. The routing request can either be get request or post request. This can be handled by the router by specifying the method. The router method takes two arguments 1. URL and the 2. The controller that handles the request.

There are separate controllers for Registered Hosts, Registered Users, Non-Registered Users and Admin. Each controller contains a middleware implementation which is configured in the routers. The Business logic is implemented in the controller. The controller communicates with the Model. The Model manipulates the data and gives the result to the controllers and the controllers then passes the data to View to render the data. This separation of concern is demonstrated below

One such example of Controller is. The postPaidCancellation Controller takes three arguments. The request, response and next object. This controller handles the paid cancellation of the booking by the Registered User.

exports.postPaidCancellation = (req, res, next) =>{

let sess = req.session;

const booking\_id = req.body.booking\_id;

console.log(booking\_id);

const cancelation\_charge = req.body.cancellation\_charge;

Bookings.deleteBooking(booking\_id).then(result =>{

const user\_name = sess.userCredentials.user\_name;

const user\_id = sess.userCredentials.\_id; *// this is of type string*

let aux\_array = [];

*// Set your secret key: remember to change this to your live secret key in production*

*// See your keys here: https://dashboard.stripe.com/account/apikeys*

var stripe = require("stripe")("sk\_test\_6SY2LcDXwkLcXil0lCEIFvXq005Xa3W26F");

*// Token is created using Checkout or Elements!*

*// Get the payment token ID submitted by the form:*

const token = req.body.stripeToken; *// Using Express*

(async () => {

const charge = *await* stripe.charges.create({

amount: req.body.cancellation\_charge \* 100,

currency: 'usd',

description: 'Cancellation Charge',

source: token,

metadata : {Booking\_id\_canceled : req.body.booking\_id.toString(), User\_who\_canceled\_this\_property : sess.userCredentials.user\_name}

}) *// charge ends*

Bookings.fetchCurrentlyBookedUserProperties(user\_id).then(currentlyBookedProperties =>{

*if*(currentlyBookedProperties.length === 0){

*// if there are no bookings of this user in the bookings table*

res.render('registered-users/view-booked-properties',{

a : [],

b : [],

username : user\_name,

message : false,

delete\_message : true

}); *// render ends here*

} *// if ends here*

*else* {

currentlyBookedProperties.forEach( (element) =>{

Bookings.fetchPropertyDetailsFromhostProperty(element.host\_property\_id).then(ans =>{

*if*(aux\_array.length < currentlyBookedProperties.length){

aux\_array.push(ans)

*if*(aux\_array.length === currentlyBookedProperties.length){

*//console.log('AUX ARRAY IS ')*

*//console.log(aux\_array);*

*//console.log('B IS ')*

*//console.log(aux\_array);*

res.render('registered-users/view-booked-properties',{

a : aux\_array,

b : currentlyBookedProperties,

username : user\_name,

message : false,

delete\_message : true

}) *// render ends*

} *// inner if ends*

} *// if ends*

}).catch(err =>{

console.log(err);

}) *// Bookings.fetchPropertyDetailsFromhostProperty promise ends*

}) *// forEach ends*

} *// else ends here*

}).catch(err =>{

console.log('error in fetching host properties from bookings')

}) *// Bookings.fetchCurrentlyBookedHostProperties promise over*

})();

}).catch(err =>{

console.log(err);

})

}

The res,render() renders the view. This method takes two arguments. The path of the EJS template which is to be rendered and the javascript object which contains the data to be passed to the templating engine (EJS View).

Inside the EJS view, the passed data can be embedded to the html by the wrapping the scriptlet tags <% = objectKey%>.

One such demo EJS view is view-booked-properties.ejs which displays the currently booked properties by the User.

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<meta *http-equiv*="X-UA-Compatible" *content*="ie=edge">

<link *rel*="stylesheet" *href*="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<title>Document</title>

</head>

<body>

<% if (delete\_message === true) { %>

<div align="center">

<div *class*="alert alert-success">

<strong>Booking Successfully Deleted!</strong>

</div>

</div>

<% } %>

<nav *class*="navbar navbar-expand-sm navbar-light bg-light">

<a *class*="navbar-brand" *href*="/">BookWithMe</a>

<div *class*='container'>

<ul *class*="navbar-nav mr-auto">

<li *class*="nav-item active">

<a *class*="nav-link" *href*="manage-credentials">Manage your Credentials

<span *class*="sr-only">(current)</span>

</a>

</li>

<li *class*="nav-item active">

<a *class*="nav-link" *href*="/manage-booking">Manage Bookings

<span *class*="sr-only">(current)</span>

</a>

</li>

<li *class*="nav-item active">

<a *class*="nav-link" *href*="/book-properties-nav">Explore and Book Properties

<span *class*="sr-only">(current)</span>

</a>

</li>

<li *class*="nav-item active">

<a *class*="nav-link" *href*="/previous-bookings">Show My Previous Bookings

<span *class*="sr-only">(current)</span>

</a>

</li>

</ul>

</div>

</nav>

<div *class*="jumbotron" align="center">

<h1 *class*="display-4">Hello, <%= username %></h1>

<p *class*="lead">View properties booked by you.</p>

<hr *class*="my-4">

<p>We have your currently booked properties and also maintain properties booked by you in near past.</p>

<p *class*="lead">

</p>

</div>

<br>

<br>

</div>

<div *class*="container" align='center'>

<div align='center`'>

<p *class*='lead display-4'>

Currently Booked Properties are

</p>

<br>

<br>

</div>

<%if (a.length === 0) { %>

<div *class*="container">

<p *class*="lead">

You have not booked any properties.

</p>

</div>

<% } else { %>

<% for(var i = 0; i < a.length; i++ ) { %>

<div *class*="card bg-secondary text-white" align="center">

<div *class*="card-header" *id*="id1">

<p *class*="h3">

<%= a[i].property\_name%>

</p>

</div>

<div *class*="card-body">

<h5 *class*="card-title"><%= a[i].property\_name%> %></h5>

<p *class*="card-text"><%= a[i].address%></p>

<p *class*="display-5"> You booked this property on <%= b[i].date\_when\_property\_booked %> from <%= b[i].check\_in\_date %> to <%= b[i].check\_out\_date %></p>

<div *class*="center">

<div *class*="container">

<div *class*="row">

<div *class*="col-sm">

<form *action*="/property-details" *method* ="POST">

<input *type*="hidden" *name*="property\_id" *value*="<%= a[i].\_id%>">

<button *type*="submit" *class*="btn btn-info">View Property Details</button>

</form>

</div>

<div *class*="col-sm">

<form *action*="/rate-property" *method*="POST">

<input *type*="hidden" *value*="<%= a[i].\_id%>" *name*="property\_id">

<input *type*="hidden" *name*="chk\_in\_date\_from\_booking" *value*="<%= b[i].check\_in\_date %>">

<input *type*="hidden" *name*="chk\_out\_date\_from\_booking" *value*="<%= b[i].check\_out\_date %>">

<input *type*="hidden" *name*="bookings\_database\_id" *value*="<%= b[i].\_id %>">

<button *type*="submit" *class*="btn btn-info">Rate this property</button>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<br>

<br>

<% } %>

<% } %>

<script *src*="https://code.jquery.com/jquery-3.3.1.slim.min.js" *integrity*="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"

*crossorigin*="anonymous"></script>

<script *src*="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" *integrity*="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"

*crossorigin*="anonymous"></script>

<script *src*="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js" *integrity*="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T"

*crossorigin*="anonymous"></script>

</body>

</body>

</html>

The ***.gitignore*** file contains the folder that are not to be tracked by the git version controlling system.

The ***package.json*** contains the metadata for the project package. The package.json file also contains the information about the ownership and license of the project. The runtime configuration scripts are specified in the package.json file. Also the file contain the version of all the third party packages used in the project. The package.json is generate by the Node’s package manager. The package.json file for this project is

{

"name": "bookwithme",

"version": "1.0.0",

"description": "bookwothme is an online rental service",

"main": "app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "nodemon app.js"

},

"repository": {

"type": "git",

"url": "git+https://github.com/PrayagDesaiweb/BookWithMe.git"

},

"author": "Prayag Desai",

"license": "ISC",

"bugs": {

"url": "https://github.com/PrayagDesaiweb/BookWithMe/issues"

},

"homepage": "https://github.com/PrayagDesaiweb/BookWithMe#readme",

"dependencies": {

"body-parser": "^1.18.3",

"bootstrap": "^4.2.1",

"cookie-parser": "^1.4.3",

"ejs": "^2.6.1",

"express": "^4.16.4",

"express-session": "^1.15.6",

"jquery": "^3.3.1",

"mongodb": "^3.1.11",

"node-schedule": "^1.3.2",

"nodemon": "^1.18.9",

"popper.js": "^1.14.7",

"stripe": "^6.28.0"

}

}

In this project following third-party Node Packages are used

"dependencies": {

"body-parser": "^1.18.3",

"bootstrap": "^4.2.1",

"cookie-parser": "^1.4.3",

"ejs": "^2.6.1",

"express": "^4.16.4",

"express-session": "^1.15.6",

"jquery": "^3.3.1",

"mongodb": "^3.1.11",

"node-schedule": "^1.3.2",

"nodemon": "^1.18.9",

"popper.js": "^1.14.7",

"stripe": "^6.28.0"

}

The projects needs this packages for development of the application. All this depenedency packages are located inside the ***node modules*** folder.

**5.5 Coding Standards**

Coding standards for development for Nodejs Express are strictly followed. The naming conventions are strictly adhered by. All the code are check for exceptions and all the exceptions are handled. Async code is handled by promises so that the code is made error free, accurate as much as possible. Latest JavaScript ES6 features are implemented in the project. Also the code contains relevant documentation.

**Chapter 6**

**Test Case Design**

**7.1 Test Cases**

This system is a conglomeration of three main components. Registered Hosts, Registered Users and Non-Registered-Users. Each f these test cases are discussed.

**7.1.1 Test case for Non-Registered Users**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case Id** | **Test**  **Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Expected Result** | **Pass/**  **Fail** |
| NRU-1 | Check navigation to user-login | Click user-login in the navbar` | - | Directed to the user-login page | Directed to the user-login page | Pass |
| NRU-2 | Check navigation to host login | Click host-login in the navbar` | - | Directed to the host-login page | Directed to the host-login page | Pass |
| NRU-3 | Check navigation to user-signup | Click user-signup  in the navbar` | - | Directed to the user-sign-up page | Directed to the user-signup page | Pass |
| NRU-4 | Check navigation to host-signup | Click host-signup in the navbar` | - | Directed to the host-signup page | Directed to the host-signup page | Pass |
| NRU-5 | Check accommodation  Or explore properties as per location and check-in credentials | Fill in the information on where and when you want to view properties` | City= Nadiad  State=Gujarat  Check-in-date: 28-4-19  Check-out-date : 1- 3 - 19 | Page shows properties in Nadiad according to the data entered but the non-regsistered users | Page shows properties in Nadiad according to the data entered but the non-regsistered users | Pass |

**7.1.1 Test case for Registered Users**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case Id** | **Test**  **Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Expected Result** | **Pass/**  **Fail** |
| RU-1 | Log-in as Registered user | Enter the username and password | Username = Prayagdesai,  Password =  Password | Directed to the User dashboard | Directed to host dashBoard | Pass |
| RU-2 | Log-in as Registered User | Enter the username and password | - Username = Prayagdesai,  Password =  Password12 | Redirected to the host-login page. User Credentials are incorrect. | Redirected to the host-login page. | Pass |
| RU-3 | Update credentials | Enter the changed credentials | Username =  Prayagdesai1234 | Redirected to log-in page to log-in with the new credentials | Redirected to log-in page to log-in with the new credentials | Pass |
| RU-4 | Manage Bookings | View Rental details, Cancel this booking or view booking details | - | If current bookings are made then the bookings will be shown. Else System will show no bookings | If current bookings are made then the bookings will be shown. Else System will show no bookings | Pass |
| RU-5 | View property details | Navigate to the Manage Bookings and click on the view property detail link button of the booking. | - | The page shows the property details and all the information regarding the property | The page shows the property details and all the information regarding the property | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case Id** | **Test**  **Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Expected Result** | **Pass/**  **Fail** |
| RU-6 | Cancel this booking | Navigate to the Manage Bookings and click on the view cancel this booking  button of the booking. | - | If the booking is under free-cancellation scheme, The booking is deleted. If the booking is under paid cancellation, User is Directed to payment page to pay the cancellation charge | If the booking is under free-cancellation scheme, The booking is deleted. If the booking is under paid cancellation, User is Directed to payment page to pay the cancellation charge | Pass |
| RU-7 | Delete paid booking | Navigate to the Manage Bookings and click on the view cancel this booking  button of the booking. If the booking is under paid cancellation scheme, user will redirect to the delete-booking page | Email =  [Prd243@gmail.com](mailto:Prd243@gmail.com)  Card Number =  4242424242424242  Month and date =  09/19  CVC Number = 1203 | On successful payment of cancellation fee, User is redirected to the host-dashboard page. The user will see the deleted property is removed from the user dashboard. | On successful payment of cancellation fee, User is redirected to the host-dashboard page. The user will see the deleted property is removed from the user  dashboard | Pass |
| RU-8 | View Booking details | Navigate to the Manage Bookings and click on the view property detail link button of the booking. | - | The booking related information and detail are displayed to the User | The booking related information and detail are displayed to  The User | Pass |
| RU-9 | Explore Properties by location | Navigate to Book and Explore properties from the User Dashboard | City = Dallas  State = Texas | The System lists all the property available for rent in Dallas, Texas. The system will show no property if there are no property in the specifies city and state | The System lists all the property available for rent in Dallas, Texas The system will show no property if there are no property in the specifies city and state | Pass |
| RU-10 | Explore property by location and check-in and check-out date and number of accommodation per rental | Navigate to the Manage Bookings and click on the view property detail link button of the property. | City= Dallas  State = Texas  Check-in-date =  12-10-2019  Check-out-date =  22-10-2019  Accommodation per rental = 5 | The System lists all the property available for rent in Dallas, Texas. The system will show no property if there are no property in the specifies city and state | The System lists all the property available for rent in Dallas, Texas. The system will show no property if there are no property in the specifies city and state | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RU-11 | View only BookWithMe properties | Navigate to the Manage Bookings and click on the view property detail link button of the booking. If there are properties in the specified location, Click on the View only BookWithMe plus properties | - | The System will show only BookWithMe plus properties to the User. If there are no such properties, the system will show no properties to the user | The System will show only BookWithMe plus properties to the User. If there are no such properties, the system will show no properties to the user | Pass |
| RU-12 | View only BookWithMe Entire Homes | Navigate to the Manage Bookings and click on the view property detail link button of the booking. If there are properties in the specified location, Click on the View only BookWithMe entire Homes | - | The System will show only BookWithMe Entire Homes to the User. If there are no such properties, the system will show no properties to the user | The System will show only BookWithMe Entire Home to the User. If there are no such properties, the system will show no properties to the user | Pass |
| RU-13 | View only BookWithMe  Boutique Rooms | Navigate to the Manage Bookings and click on the view property detail link button of the booking. If there are properties in the specified location, Click on the View only BookWithMe Boutique rooms | - | The System will show only BookWithMe Boutique rooms properties to the User. If there are no such properties, the system will show no properties to the user | The System will show only BookWithMe boutique rooms properties to the User. If there are no such properties, the system will show no properties to the user | Pass |
| RU-14 | Rate the property | Navigate to user dashboard and select the property you want to give rating to | Enter the details about your experience in the property, Add review about the property and the Host’s hospitality. Also give final rating. | The system will submit the rating given bu the user. If the user has not stayed in the property after the booking. The User then cannot rate the property | The system will submit the rating given bu the user. If the user has not stayed in the property after the booking. The User then cannot rate the property | Pass |
| RU-15 | View Previously booked properties. | Navigate to View previously booked properties | - | The system will show you all your previous bookings if you have any. Or else the system will show you no previous bookings made by the User | The system will show you all your previous bookings if you have any. Or else the system will show you no previous bookings made by the User | Pass |

**Chapter 7**

**Conclusions and Future Extension**

## Conclusion

This web application provides a very clean and effective solutions for user to book properties and host to rent their properties using the platform. This application also targets clean design of displaying and booking properties to its users, and the UI is design such that the users will not feel hesitated or confused to use this application. Also, there is a powerful and feature rich UI is designed and functionalities are implemented so that the hosts are easily and flexibly able to rent their properties to the users without any hustle. Bookings and Check-in and Check-out is implemented in such an efficient fashion which makes this application very effective to use.

## Discussion

### **Self-Analysis of Project Viabilities**

According to us, this project is completed with the primary functionalities as specified earlier but then again there is lot more than this which can be done. The project is well capable to handle the given job for some particular task but not all of them. So then it is a challenge to further develop it in to a well flagged software as it was challenge to develop up to this stage.

### **Problem Encountered while development**

There were many problems encountered during the design and the development phase of the project.

* Problem regarding web application layout.
* Problem to maintain database.
* Problem regarding multiple synchronous requests to the server.
* Problem regarding handling Async code in Node.js
* Problem regarding parsing some JavaScript content in EJS templating engine.
* Problem in deciding the deadline to implement certain features in the project

### **Summary of Project Work**

I have completed our project work using software engineering and system analysis and design approach. We have considered all the suggestions given by my internal guide and have also followed the desired instructions on documentation and coding standards stated by the project guides. I have done work with pre-planned scheduling related with time constraints and result oriented progress in project development.

**7.3 .** **Limitations**

The System lack Authorization of the admin, hosts and users. Also, the System lacks feature for uploading images by the host of their property. The system lacks mechanism to suggest user new properties to book and perform analytics on the user generated data.

* 1. **Future Extension**

There are some functionalities that can be implemented later which can be very important besides core functionalities that are implemented in this project. This project can be further extended by adding following functionalities

* Implementing Recommendation System which recommends user the property that suits the taste for the user based on the user’s previous bookings.
* Implementation of google analytics so that the platform come to knows the trends for booking and hosting of the properties across several locations.
* Image uploading feature by Registered Hosts so that the users can have a view of the property that they have to book.
* Enhancement in the front-end and UX and UI design

**7.5 Bibliography**

* **JavaScript:** <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
* **Node.js:** <https://nodejs.org/en/docs/>
* **Express:** <https://expressjs.com/en/api.html>
* **Bootstrap:** <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
* **CSS:** <https://www.w3schools.com/w3css/>
* **jQuery:** <https://api.jquery.com/>
* **EJS:** <https://ejs.co/>
* **Stripe Online Payment Gateway:** <https://stripe.com/docs/quickstart>
* **MongoDB:** <https://docs.mongodb.com/>