*Project report*

*on*

*Rent hive*



## DAV

## University

DAV University

For the partial fulfilment of the requirement for the award of degree of

“BACHELOR OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY”

(2022-2026)

Submitted To: Submitted By:

(Teacher’s Name) Deepjot Kaur

**DECLARATION**

*I,* ***Deepjot Kaur****, hereby declare that the project report entitled* ***“Rent Hive”*** *submitted by me to* ***DAV University****, in partial fulfilment of the requirements for the award of the degree of* ***Bachelor of Computer Science and Information Technology (Semester IV)****, is my original work and has not been submitted previously to any other university or institution for the award of any degree or diploma.  
I further declare that any material or ideas borrowed from other sources have been duly acknowledged and referenced in this project report*

Place: DAV UNIVERSITY

Signature of the Student:

**Acknowledgement**

I have received invaluable support and assistance in preparing this project. While a brief acknowledgment here in no way repays their contributions, it is a

small courtesy expressed with sincere gratitude.

I would like to extend my heartfelt thanks to all the individuals who helped me in various ways in the development of this project report. Without their continuous support and guidance, the completion of this project would no

have been possible.

I extend my deep gratitude to **Mr. Arvind mahindru ,** Head of the Department, School of Computational Science, for his constant encouragement and support

throughout my degree.

I also wish to express my sincere thanks to my supervisor, **[Supervisor’s Name]**, for her invaluable guidance, advice, and unwavering support. I will always carry her guidance with me throughout my life.

I would like to place on record my sincere thanks to all those writers and authors whose works have been a great source of knowledge and inspiration for me.

I express my special gratitude to the **O7 Services Staff** for giving me their valuable time and attention.

Finally, I would like to express my deepest thanks to my parents for their continuous support, love, and motivation, which gave me the strength and determination to succeed. My sincere appreciation also extends to all those who willingly helped me in any capacity during the completion of this project.

Deepjot Kaur

**CHAPTER – 1 INTRODUCTION**

### **1.1 INTRODUCTION**

In today’s fast-paced world, convenience, transparency, and trust play a vital role in accommodation services, especially for students and working professionals seeking Paying Guest (PG) facilities. **“Rent Hive”** is designed to simplify and modernize the PG renting experience by providing a single digital platform that connects PG owners, users, and admins seamlessly.

This project aims to bridge the gap between offline PG searching hassles and a streamlined online booking system. **Rent Hive** offers verified PG listings, real-time room availability, secure online bookings, and city-wise filtering — eliminating the dependency on brokers and saving users valuable time and money.

The platform comprises three main modules: **Admin**, **PG Owners**, and **Users**.

* The **Admin module** provides powerful tools to manage and verify cities, PG owners, users, and room details, ensuring smooth system operations.
* The **PG Owner module** enables owners to easily add, update, or remove PG listings, manage room availability, and view booking details.
* The **User module** offers a user-friendly interface for users to register, log in, search PGs based on city and preferences, view verified details, check availability, and book rooms online with secure payments.

By leveraging modern web technologies, **Rent Hive** aims to provide a transparent, reliable, and efficient platform for PG renting, empowering both owners and tenants with a hassle-free experience.

**1.2 PROJECT DESCRIPTION**

**Modules of Project:**  
There are various modules associated with the project. These modules work in their specific areas to lead and complete the system successfully.

**1. Admin Module**

This module equips administrators with comprehensive tools to manage the system efficiently. The functionalities include:

* **Admin Login:** Secure access to the admin panel.
* **Manage Cities:** Add, update, or delete city details for PG listings.
* **Manage PG Owners:** Approve or reject owner registrations, update details.
* **Manage Users:** View and manage registered users’ information.
* **Manage Rooms:** Verify and monitor PG listings and room availability.
* **Logout:** Securely exit the admin panel.

#### **2. PG Owner Module**

This module allows PG owners to manage their listings easily. The functionalities include:

* **Owner Login:** Secure access for PG owners.
* **Manage PG Details:** Add, update, or delete PG information.
* **Manage Rooms:** Update room availability status.
* **View Bookings:** Check booking details and user information.
* **Logout:** Securely log out from the owner panel.

#### **3. User Module**

This module provides a simple and interactive interface for users. The functionalities include:

* **User Registration & Login:** Sign up and securely log in to the system.
* **Search PGs:** Search PGs based on city, area, and filters.
* **View Listings:** View verified PG details, room availability, and amenities.
* **Book Rooms:** Book rooms online with secure payment options.
* **View Booking History:** Track booking status and history.
* **Manage Profile:** Update personal details.
* **Logout:** Securely log out from the user account.

**1.3 PROBLEM DEFINITION**

Previous PG renting systems face several challenges that hinder efficiency, transparency, and trust. These problems include:

* **Manual Administrative Tasks:** Traditional PG management involves time-consuming and error-prone manual work such as maintaining booking records and updating room availability.
* **Limited Accessibility and Information:** Users often struggle to find verified information about PGs, owners, and room availability, making it hard to take informed decisions.
* **Inefficient Booking Process:** The absence of a centralized online system leads to delays in booking confirmation and room status updates, resulting in confusion and overbooking.
* **Dependency on Brokers:** Users often rely heavily on brokers who charge extra commissions and may provide unreliable information.
* **Lack of Customer Feedback:** Traditional systems do not have effective ways to collect and utilize user feedback, affecting service improvements and trust.

## ****1.4 EXISTING SYSTEM****

The current PG renting process still relies on outdated practices and manual processes, including:

* **Physical Visits & Paperwork:** Tenants have to visit multiple PGs physically and rely on paperwork or word-of-mouth, wasting time and effort.
* **No Centralized Platform:** There is no single verified platform that shows real-time availability, amenities, and owner details.
* **Limited User-Friendly Interface:** The process is not digitalized or user-friendly, making it difficult for users to compare options and book efficiently.
* **Inefficient Administrative Tools:** PG owners manually handle bookings, room status, and payments, which is prone to human error and miscommunication.

## ****1.5 PROPOSED SYSTEM****

The proposed system, **“Rent Hive”**, aims to address these challenges by developing an advanced, secure, and user-friendly web-based platform for PG renting.

**Key features of the proposed system include:**

**Admin Module:**

* Provides admins with a centralized dashboard to manage cities, PG owners, users, and room listings.
* Enables verification of owner details and PG listings to maintain trust and quality.
* Monitors bookings and ensures real-time data accuracy.

**PG Owner Module:**

* Allows PG owners to easily add, update, or delete PG listings and manage room availability.
* Helps owners track bookings and user details in real-time, reducing manual errors.

**User Module:**

* Provides users with a seamless interface to register, search PGs city-wise, view verified information, and book rooms online securely.
* Offers real-time availability, transparent pricing, and secure payment gateways.
* Enables users to manage their bookings and give feedback, improving overall service quality.

By implementing **Rent Hive**, the entire PG renting process becomes simpler, more transparent, and efficient — eliminating broker dependency and ensuring a hassle-free experience for both users and PG owners.

**CHAPTER-2 Hardware & Software Requirements**

For this project minimum hardware and software requirement are listed below:

HARDWARE REQUIREMENTS

● Processor: Intel Core i3 or higher (e.g., Intel Core i5, Intel Core i7)

● Ram: 8 GB

● SSD:256GB

SOFTWARE REQUIREMENTS

● Front End: HTML, CSS, Bootstrap, JavaScript, ECMA Script, React JS

● DB Tool: Firebase Fire store

● Browser: Mozilla Firefox/Chrome/Edge or any other relevant browser

● OS: Windows operating system/Linux

● Text Editor: Visual Studio

## ****CHAPTER – 3 FEASIBILITY STUDY****

The **Rent Hive PG Renting System** aims to provide a seamless digital platform for students and working professionals to find, compare, and book verified PG accommodations easily. It enhances the renting experience through real-time information, secure bookings, and efficient management for owners and admins.

**3.1 ECONOMIC FEASIBILITY**

The economic feasibility of the **Rent Hive PG Renting System** involves assessing its financial viability and long-term sustainability. The platform is free for users to access and allows PG owners to list and manage their PG details easily through the provided interface.

The increasing demand for affordable and verified PG accommodations, especially among students and working professionals migrating to cities, shows significant potential. A competitive analysis of existing brokers and local listing services highlights how **Rent Hive** can reduce dependency on costly intermediaries and hidden charges.

By digitizing the entire process and eliminating manual tasks and broker commissions, **Rent Hive** ensures cost-effectiveness for both tenants and PG owners, making the system economically viable and sustainable.

### **3.2 TECHNICAL FEASIBILITY**

The system will be built using modern web technologies to ensure a responsive and user-friendly experience.

* **Frontend:** HTML, CSS, JavaScript (or react) and Bootstrap will be used to create a clean and responsive interface, making it easy for users to search PGs and make bookings.
* **Backend:** Firebase will be used for real-time database management, authentication, and secure hosting. Its real-time capabilities make it ideal for updating room availability instantly.
* The chosen tech stack is lightweight, scalable, and well-supported, ensuring the platform runs smoothly and can handle increasing user demands.

### **3.3 BEHAVIOURAL FEASIBILITY**

The system features an attractive and intuitive user interface built with HTML, CSS, JavaScript, Bootstrap, and React, ensuring ease of use for both tenants and PG owners. Verified listings, real-time availability, secure online payments, and city-wise filtering promote trust and convenience.

**Rent Hive** aims to provide a seamless and transparent user experience with features like:

* Real-time room status updates
* Easy and secure online bookings
* Feedback mechanisms for users to share reviews and suggestions

Continuous support, regular updates, and user feedback incorporation will ensure high satisfaction and adoption rates.

### **3.4 METHODOLOGY / PLANNING OF WORK**

The main objectives and work plan for the **Rent Hive PG Renting System** are:

* Define clear objectives and scope, and plan tasks and timelines for efficient development.
* Use HTML, CSS, Bootstrap, and React for frontend development.
* Implement Firebase for backend data storage, authentication, and real-time updates.
* Conduct thorough testing of all modules (Admin, PG Owner, User) to ensure accuracy and reliability.
* Regularly monitor project progress, gather user feedback, and make improvements to enhance system usability and performance.

.

## ****CHAPTER 4 SYSTEM ANALYSIS****

**4.1 Data Analysis**  
Before developing the Rent Hive PG Renting System, we studied the existing manual PG renting process. The system is analysed from **operational**, **technical**, and **economic** perspectives to check its practical implementation.

**Types of Analysis:**

* **Operational Analysis:** Checks if the system will work smoothly for owners, admins, and users.
* **Technical Analysis:** Checks if the technology used is secure, scalable, and user-friendly.
* **Economic Analysis:** Checks if the cost of making and running the system is worth the benefits.

**4.2 Operational Analysis**  
The Rent Hive system will automate PG bookings, update room status in real-time, and manage owner-user interactions digitally. It will reduce broker dependency, human errors, and paperwork.  
Features like secure login, backups, and simple interfaces will make daily operations easy, safe, and user-friendly.

**4.3 Technical Analysis**  
Technically, the system is feasible because:

* Frontend: HTML, CSS, JS, Bootstrap, React = responsive design.
* Backend: Firebase = real-time database, secure authentication, and cloud storage.
* Data backups keep information safe.
* System can handle many users at a time.

**4.4 Economic Analysis**  
The system will have basic development costs for designing and hosting the website.  
No expensive hardware is needed because everything is cloud-based.  
In the long term, Rent Hive will save money for users and PG owners by removing broker fees, paperwork, and manual work.  
This makes it cost-effective and sustainable.

## Chapter 5:

## Technology

## ****5.1 HTML****

* **HTML** stands for **Hyper Text Markup Language**.
* It is the standard markup language for creating web pages.
* HTML describes the structure of a web page using various elements and tags.
* HTML elements tell the browser how to display the content.
* HTML files are the backbone of any website and can be enhanced with CSS and JavaScript.

**Explanation:**  
Hyper Text Markup Language (**HTML**) defines the content and structure of web documents. In the **Rent Hive PG Renting System**, HTML is used to create the basic layout of pages like the home page, login/signup forms, PG listing pages, booking forms, and admin panels.  
It works together with **CSS** for styling and **JavaScript** or **React** for dynamic interactivity.

Web browsers receive HTML files from the server or cloud storage and render them as user-friendly web pages. Thus, HTML is the foundation of our project’s frontend design.



### Advantages:

* HTML helps build the basic **structure** of a website and is the most widely used markup language.
* It is easy to learn and implement.
* Every browser supports HTML, making it highly compatible.
* HTML is lightweight and fast to load.

### Disadvantages:

* It cannot produce dynamic output alone, since it’s a static language.
* Managing the structure of large HTML documents can be difficult to understand.
* Errors can cause big issues and may be hard to debug.
* It can be time-consuming to maintain the design and layout consistency for big projects.

## ****5.2 CSS****

* **CSS** stands for **Cascading Style Sheets**.
* CSS defines how HTML elements should appear on screen, paper, or other media.
* CSS saves a lot of work by controlling the layout of multiple pages from one file.
* External stylesheets are stored in **.css** files and linked to HTML documents.

CSS is used to style and format HTML documents, improving their visual appearance. It allows changes in fonts, colours, margins, spacing, layouts, and even animations.  
In the **Rent Hive system**, CSS ensures that all pages like room listings, owner dashboards, and user profiles look clean, attractive, and consistent.

**Ways to use CSS:**

* **Inline CSS:** Styles written directly in HTML elements.
* **Internal CSS:** Styles written inside <style> tags in the HTML <head>.
* **External CSS:** Styles written in separate.css files linked to the HTML, making the design reusable and easier to maintain.



## Advantages of CSS:

* CSS plays an important role, by using CSS you simply got to specify a repeated style

for element once &amp; use it multiple times as because CSS will automatically apply the

required styles.

* The main advantage of CSS is that style is applied consistently across variety of sites.

One instruction can control several areas which is advantageous.

* Web designers needs to use few lines of programming for every page improving site

speed.

* Cascading sheet not only simplifies website development, but also simplifies the

maintenance as a change of one line of code affects the whole web site and

maintenance time.

## Disadvantages of CSS:

* Browser compatibility (some styles sheets are supported and some are not).
* CSS works differently on different browsers. IE and Opera supports CSS as different

logic.

* There might be cross-browser issues while using CSS.
* There are multiple levels which creates confusion for non-developers and beginners.

## ****5.3 JavaScript****

JavaScript (**JS**) is a lightweight, object-oriented scripting language used to make webpages dynamic and interactive.  
It is a full-fledged programming language that runs in the browser and allows users to interact with webpages without needing to reload them every time.

JavaScript was introduced in 1995 with the Netscape Navigator browser and is now supported by all major web browsers.

JavaScript can be added to HTML in two ways:

* **Internal JS:** Code is written inside the <script> tag, placed in the <head> or <body> section of the HTML file.
* **External JS:** Code is written in a separatee.js file and linked to the HTML file using the <script> tag.

### **Applications of JavaScript:**

### JavaScript is used to add interactivity and dynamic features to webpages, such as:

* Validating user input on forms.
* Creating dynamic drop-down menus.
* Displaying date, time, or clocks.
* Showing pop-up windows, alerts, confirm or prompt dialog boxes.
* Adding animations or interactive content.

### **Use in Rent Hive:**

### In the **Rent Hive PG Renting System**, JavaScript is used to:

* Validate user registration and login forms.
* Show real-time updates (like availability of rooms).
* Provide dynamic filtering for PG search results.
* Improve user experience with interactive features



**5.4 Bootstrap**

In this project, **Bootstrap** is used as the front-end framework to simplify and speed up the design and development process. Bootstrap is an HTML, CSS, and JS library that helps developers create responsive and uniform web pages with minimal effort. By adding Bootstrap to this project, I have applied its predefined styles for colours, sizes, fonts, and layouts to ensure a clean and consistent look across all browsers and devices.

It provides basic style definitions for elements like text, tables, forms, and buttons, which helps maintain visual consistency. In addition, I have used Bootstrap’s CSS classes and responsive grid system to arrange the content properly on different screen sizes.

Bootstrap’s JavaScript components, such as **modals**, **tooltips**, and **navigation bars**, have also been used to enhance the user interface without needing extra libraries like jQuery. Overall, Bootstrap has played an important role in making the project **responsive, attractive, and user-friendly**.

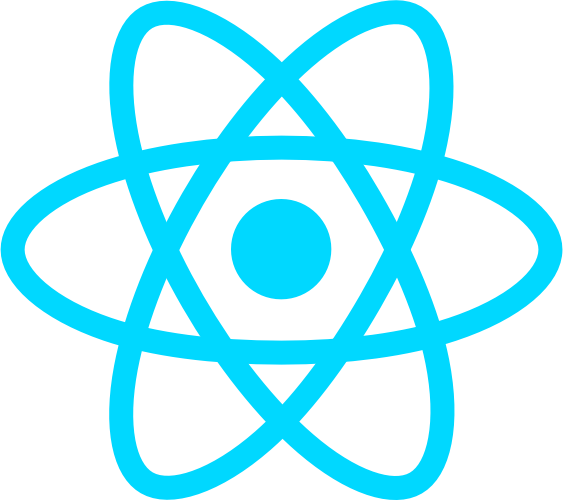


5.5 React

In this project, **react** has been used as the front-end library to build a dynamic and interactive user interface. React (also known as React.js or ReactJS) is a free and open-source JavaScript library maintained by Meta (formerly Facebook) and a large community.

React allows the development of **single-page applications** by breaking the UI into **reusable components**. Each component manages its own state and uses **props** to pass data between components. This makes the application modular, easier to debug, and scalable.

One of the main advantages of React is its **virtual DOM**, which ensures that only the parts of the web page that change are re-rendered, improving performance. In this project, react has been used to render different pages dynamically without reloading, handle routing (with libraries like React Router if used), and manage the state efficiently.



[This Photo](https://damiandeluca.com.ar/5-caracteristicas-de-react-que-deberias-conocer) by Unknown Author is licensed under [CC BY-NC-ND](https://creativecommons.org/licenses/by-nc-nd/3.0/)

* Virtual DOM: React uses a virtual DOM to efficiently update the browser DOM. It

compares the virtual DOM with the real DOM and only applies the necessary

changes, which improves performance.

* JSX (JavaScript XML): JSX is a syntax extension in React that allows you to write

HTML-like code directly in JavaScript. This makes it easier to create and manage

React components within your codebase.

* State Management: React components can manage their own state using use State

hook (for functional components) or class-based state (for class components). This

allows components to store and update data internally, facilitating dynamic user

interfaces.

* Lifecycle Methods: Class components in React have lifecycle methods such as

Component Did Mount, component did Update, and component Will Unmount, which

allow developers to execute code at specific points during a component&#39;s lifecycle.

* React Hooks: Introduced in React 16.8, hooks are functions that let you use state and

other React features without writing a class. Hooks provide a more functional

approach to state management and side-effects in functional components.

* Component Reusability: React promotes component reusability and modularity.

Components can be composed together to build complex UIs, and they can be reused

across different parts of the application, leading to cleaner and more maintainable

code.

* Community and Ecosystem: React has a vibrant community and a rich ecosystem of

libraries, tools, and extensions (like Redux for state management, React Router for

routing, and Axios for HTTP requests) that enhance its capabilities and support

development across various use cases.

### **5.6 Firebase**

**Firebase** is a powerful **Backend-as-a-Service (BaaS)** platform owned by Google.  
It provides a wide range of **cloud-based services** that help developers build, deploy, and scale web and mobile applications quickly, without managing complex backend infrastructure. Firebase simplifies backend tasks like **real-time databases, authentication, hosting, cloud storage, messaging**, and more — all with easy-to-use SDKs and robust security.

In this project, Firebase has been used extensively for **real-time data storage**, **user authentication**, and **hosting the web application securely**.

### **1. Firebase Realtime Database**

* A **NoSQL cloud database** that stores data in **JSON format**.
* Synchronizes data across all connected clients in **real-time**.
* Supports **offline capabilities**, so data is cached locally and syncs when the device is online again.
* 🔗 Project Use: In this project, it is used to store and sync user data like profiles, posts, or messages instantly without needing page reloads.

### **2. Cloud Fire store**

* A flexible, scalable, **NoSQL cloud database** for mobile, web, and server development.
* Allows **real-time updates**, **complex querying**, and **hierarchical data structures**.
* Automatically scales with your app’s growth.
* 🔗 Project Use: Ideal for future scalability if large datasets are to be handled.

### **3. Firebase Authentication**

* Provides **secure user sign-in** methods with easy SDKs and ready-made UI screens.
* Supports email/password, OTP, and social providers like **Google, Facebook, Twitter**.
* Handles password resets, user management, and session security.
* 🔗 Project Use: Enables user registration and login functionality, ensuring only authorized users can access restricted content.

### **4. Firebase Cloud Messaging (FCM)**

* A free, cross-platform solution for **sending push notifications** to users.
* Helps maintain user engagement with real-time alerts and updates.
* 🔗 Project Use: Can be integrated in future to send notifications about new posts, messages, or updates.

### **5. Firebase Hosting**

* Provides **fast and secure web hosting** for static and dynamic content.
* Includes a **global CDN** for low latency and **free SSL certificates** for secure connections.
* Single command deployment makes updates hassle-free.
* 🔗 Project Use: The project is deployed on Firebase Hosting, ensuring fast content delivery and secure HTTPS access.

### **6. Firebase Cloud Storage**

* Allows developers to store and serve **user-generated files** like images, audio, or videos.
* Handles **large file uploads/downloads** with robust security using Firebase Authentication.
* 🔗 Project Use: Can be used to store profile pictures, uploaded documents, or media files.

### **7. Firebase Functions**

* Provides a **serverless framework** to run backend code in response to Firebase events or HTTP requests.
* Automatically scales and removes the need to manage servers.
* 🔗 Project Use: Can be added to perform operations like sending emails on registration or processing user data.

### **8. Firebase Analytics**

* A free app measurement tool that provides deep insights into **user engagement** and app performance.
* Helps in analyses user behaviour and making **data-driven decisions**.
* 🔗 Project Use: Useful to track which features are most popular and improve the app experience.

### **9. Firebase Remote Config**

* Let’s developers update the app’s appearance or behaviour **without publishing a new version**.
* Supports **A/B testing** for comparing features.
* 🔗 Project Use: Can be used in the future for feature toggles or personalized experiences for different user segments.

### **10. Firebase Crashlytics**

* Real-time crash reporter that tracks, prioritizes, and helps fix stability issues.
* Provides detailed crash logs and breadcrumbs for troubleshooting.
* 🔗 Project Use: Useful for maintaining app stability and quick bug fixes.

### **11. Firebase Test Lab**

* A cloud-based app-testing infrastructure.
* Runs automated tests on real physical and virtual devices.
* 🔗 Project Use: Ensures the web app works properly across multiple browsers and devices.

### **12. Firebase Performance Monitoring**

* Tracks the app’s performance, including **network latency** and page load times.
* Helps identify performance bottlenecks.
* 🔗 Project Use: Helps maintain a fast and responsive user experience.

### **Benefits of Using Firebase**

* **Real-time capabilities:** Data syncs instantly across clients.
* **Scalability:** Handles projects from small to large scale easily.
* **Easy to integrate:** Simple SDKs and good documentation.
* **Secure:** Built-in security rules and authentication.
* **Time-saving:** Reduces backend coding efforts.
* **Reliable:** Managed by Google, with high uptime.

### **Use Cases of Firebase**

* 🔗 **Chat Applications:** Real-time messaging and presence updates.
* 🔗 **E-commerce Platforms:** User auth, product data sync, analytics.
* 🔗 **Gaming Apps:** Real-time multiplayer sync, leader boards.
* 🔗 **Social Media Apps:** User profiles, posts, comments, push notifications.
* 🔗 **Educational Apps:** Real-time quizzes, progress tracking.

### **Firebase in This Project**

In this project, Firebase has made it possible to:

* Securely manage user registrations and logins.
* Store, update, and sync data in real-time across users.
* Deploy the web application with secure hosting.
* Plan for future integrations like push notifications, cloud functions, and advanced analytics.

Thus, Firebase provides a **complete and scalable backend solution** that saves time and makes the project robust, secure, and easy to maintain.

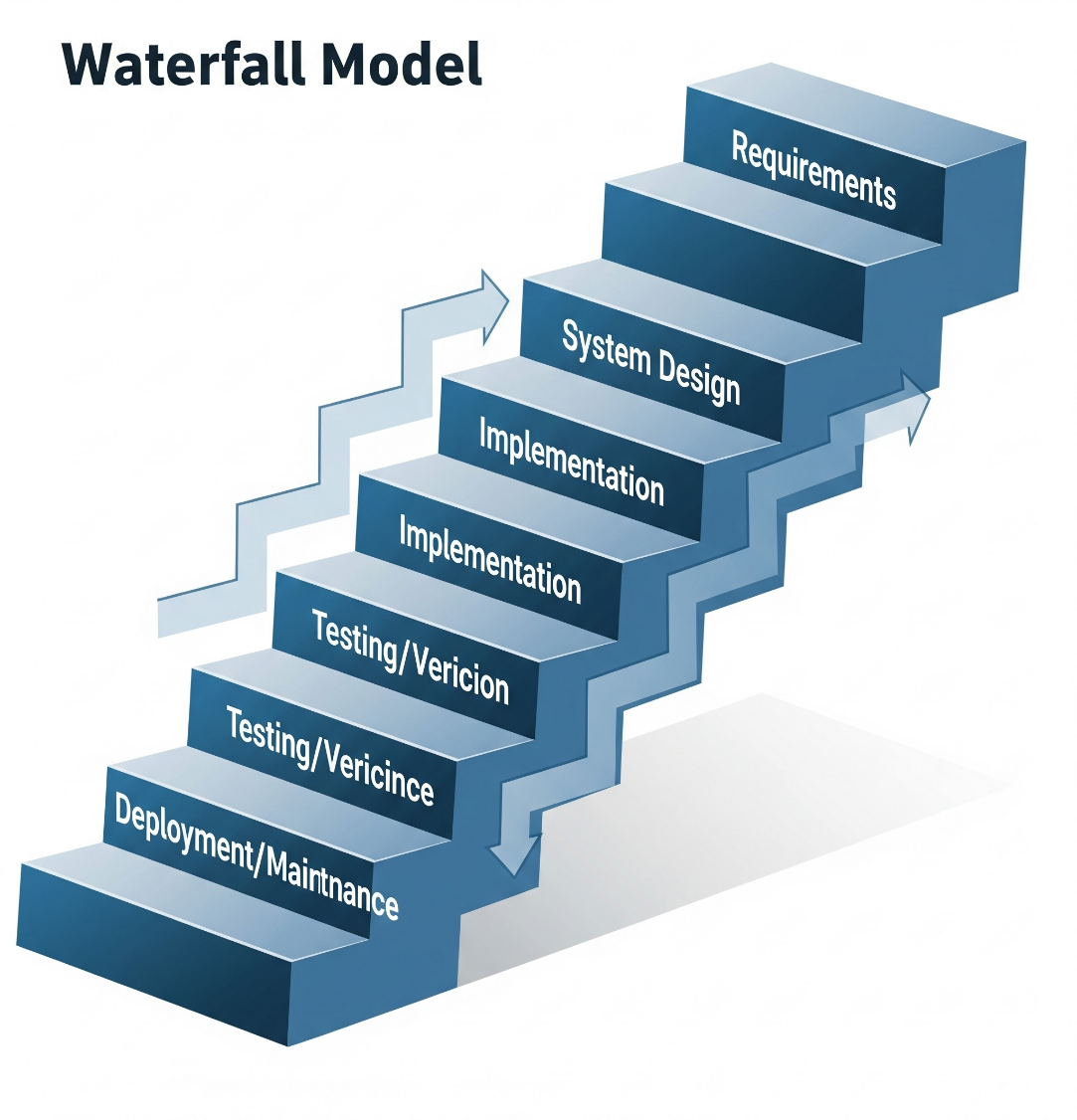
### **CHAPTER 6**

### **SOFTWARE PROCESS MODEL**

#### **Waterfall Model**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



**The sequential phases in Waterfall model are –**

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

#### Waterfall Model – Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are where:

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.

#### Waterfall Model – Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

* Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

* Simple and easy to understand and use.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.

#### Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

## ****CHAPTER 7****

## ****DESIGN****

### **7.1 System Design**

The system design phase is one of the most creative and challenging phases of the **Software Development Life Cycle (SDLC)**. The term “design” describes both the final system and the process by which it is developed. This phase involves the construction of programs and program testing. The purpose of the design phase is to plan a solution to the problem specified in the requirements document, marking the first step in transitioning from the problem domain to the solution domain. Starting with what is needed, design takes us towards how to satisfy those needs.

The design of the system is perhaps the most critical factor affecting the quality of the software, significantly impacting later phases, particularly testing and maintenance. The output of this phase is the design document, which serves as a blueprint or plan for the solution and is used during implementation, testing, and maintenance.

A systematic method is essential to achieve beneficial results at the end of the design phase. This involves starting with an initial idea and developing it into a series of steps. The series of steps for successful system development are as follows:

1. **Study the Problem:** Fully understanding the goal to be achieved is crucial. An in-depth study of the problem is the foundation of effective system design.
2. **Determine Output and Input Requirements:** Identifying the required output and the necessary input to achieve it is a very challenging but essential step in system development.
3. **Design Database Structure:** Based on the output requirements, the structure and strength of various databases should be designed to support the system efficiently.
4. **Program Development:** Deciding what kind of program to develop is crucial for reaching the final goal. This involves determining the software architecture and development approach.
5. **Write Individual Programs:** Developing individual program modules that will collectively solve the problem. These modules are later integrated.
6. **Program Testing:** Testing the individual programs and making necessary corrections to ensure they function as intended and meet the specified requirements.
7. **Integration:** Combining all the individual programs and modules, often presenting them as part of a user interface (such as a menu in a web application), to complete the software package.

The designer must keep three main objectives in mind:

1. **Performance:** How quickly and efficiently the design will perform the users' tasks given the hardware resources available.
2. **Security:** The extent to which the design is secure against human errors and machine malfunctions.
3. **Flexibility:** The ease with which the design allows the system to be modified or expanded.

To meet these objectives, analysts and programmers use two main design approaches:

### **Top-Down Design**

Top-down design, also known as system design, aims to identify the modules that should be in a system. This approach starts with a large picture and moves to the details. Analysts and team members first look at the major functions that the system must provide and then break these down into smaller and smaller activities. This method ensures that the system architecture is robust and that all necessary components are identified early in the design process.

### **Bottom-Up Design**

Bottom-up design, also known as detailed design, starts with the details and then moves to the big picture. This approach is appropriate when users have specific requirements for output. By focusing on the detailed components first, the bottom-up approach ensures that each part of the system meets the precise needs of the users before integrating them into the larger system.

Both top-down and bottom-up designs are essential for creating a well-rounded system that is both effective and adaptable. The top-down approach ensures that the overall architecture is sound and meets the system's goals, while the bottom-up approach ensures that the individual components are well-designed and functional. Combining these approaches allows for the development of a system that is both comprehensive and precise.

In the **Rent Hive PG Renting System**, the system design phase helped plan and develop different modules like **User Module**, **Admin Module**, and **Owner Dashboard** using the **top-down approach**. The database structure was designed using **Firebase** for secure storage, while the frontend layout was built with **HTML, CSS, JavaScript, and React**.  
This approach ensured that each part of the system is secure, user-friendly, and easy to maintain or expand in the future.

**CHAPTER 8**

**DFD: Data Flow Diagram**

Data Flow Diagrams (DFDs) were first developed by Larry Constantine as a method of

expressing system requirements in a graphical form. Also known as bubble charts, DFDs aim

to clarify system requirements and identify major transformations, ultimately aiding in the

system design process.

DFD is a means of representing a system at any level of detail through a graphic network of

symbols that illustrate data flows, data stores, data processes, and data sources/destinations.

**Purpose:**

The primary purpose of data flow diagrams is to provide a semantic bridge between users and

systems developers. They offer several key benefits:

 **Graphical Representation**: DFDs are graphical, which eliminates the need for

lengthy textual descriptions and makes complex systems easier to understand.

 **Logical Representation**: They model what a system does, focusing on the logical

aspect rather than the physical implementation. This helps in understanding the

functionality without getting bogged down by technical details.

** Hierarchical Structure:** DFDs are hierarchical, allowing systems to be shown at any

level of detail. This makes it easier to break down complex processes into simpler,

manageable components.

 **User Understanding and Review**: By providing a clear, visual representation of the

system, DFDs facilitate user understanding and make it easier for users to review and

provide feedback.

**DFD Symbols are as follows:**

 The External Entity symbol represents sources of data to the system or destinations of

data from the system.

 The Data Flow symbol represents the movement of data.

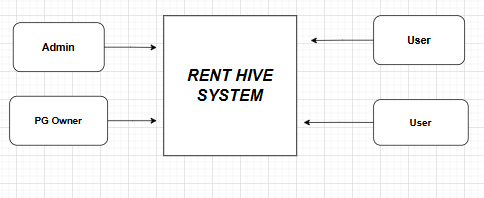
 The Data Store symbol represents data that is not moving (delayed data at rest).

 The Process symbol represents an activity that transforms or manipulates the data.

**DFD Level 0**

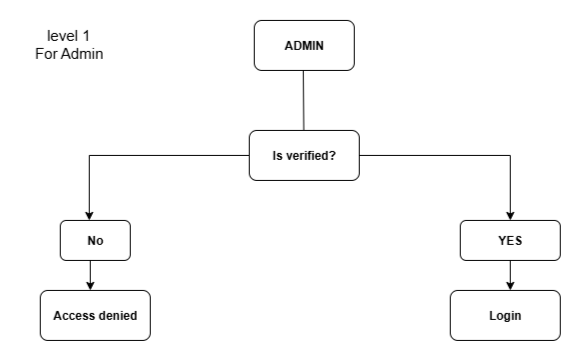
**Context Level Diagram**

“The Level 0 DFD for Rent Hive shows how Admin, PG Owners, and Users interact with the PG Renting System for managing listings, searching, booking, and payments.”

****

**Admin interacts with the Rent Hive system to manage overall operations, including verifying PG listings, managing users and owners, and monitoring bookings. Users and PG owners interact with the system for browsing available PGs, making bookings, updating listings, and giving ratings.**

**Level 1 DFD:**  
The Level 1 Data Flow Diagram for **Rent Hive** shows how the **admin** performs key processes like managing PG listings, monitoring user and owner data, handling bookings, and generating reports to ensure smooth operations.



Level 2 For user



## ****CHAPTER 10****

## ****TESTING****

### **10.1 Introduction to Testing**

Testing is the major quality control measure used during software development. It involves executing the developed modules with the intent of finding errors. No module is completely ready unless it has been thoroughly tested. This stage is crucial because it verifies whether the developed modules meet the requirement specifications defined earlier.

Testing aims to uncover any hidden errors in the system. A good test case should have a high probability of detecting undiscovered faults. If testing is performed effectively, it helps ensure that the developed software works as expected and meets all functional and non-functional requirements.

Testing is a planned activity and is carried out systematically. It is an integral part of software development, ensuring that the implemented system performs according to the user’s expectations and specifications. The main goal of testing is not just to prove that the system works correctly, but to find and fix any defects or bugs that could cause issues in real-world usage.

A common misconception is that testing is only to prove that the software works correctly, but the real objective is to detect faults. Insufficient testing can result in hidden defects, which may lead to failures when the system is deployed. Therefore, testing is a crucial step that represents the ultimate review of the specifications, design, and code. In fact, it is common for software development teams to spend up to **40%** of the total project effort on testing alone.

### **10.2 Test Strategy**

For this project, the implemented system is tested using the following basic levels of testing:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**

Each level of testing helps to detect different types of faults introduced during different phases of development:

1. **Unit Testing:**  
   This is the first level of testing. Here, each module or unit of the system is tested individually based on the design specifications. Unit testing ensures that each module’s internal logic works correctly. For this project, all individual modules were tested for correctness and performance.
2. **Integration Testing:**  
   After unit testing, modules are integrated and tested together to ensure that they work correctly when combined. The main focus is on the interfaces and interactions between integrated modules. This helps to identify errors that may occur when different modules exchange data.
3. **System Testing:**  
   Once integration testing is complete, the whole system is tested as a complete product to verify that it meets the requirements defined during the analysis phase. System testing ensures that the software behaves as expected in the intended environment and satisfies the project’s objectives.
4. **Acceptance Testing:**  
   This is the final level of testing. It is performed with real user data to validate that the system works satisfactorily in the client’s environment. Acceptance testing focuses on the external behaviour and checks whether the system fulfils the end-user requirements.

### **10.3 Test Cases**

Proper selection of test cases is essential for successful testing. There are two main approaches for selecting test cases:

* **Functional Testing:**  
  The software or module is treated as a black box. Test cases are designed based on the requirements and specifications. This method checks the external behaviour of the system without considering its internal logic. It is also known as black box testing.
* **Structural Testing:**  
  Test cases are designed based on the internal logic of the modules. The aim is to achieve maximum code coverage. A common criterion is statement coverage, which ensures that every line of code is executed at least once during testing.

### **Test Case Examples**

#### **Test Case 1: Login Screen**

* **Test Case ID:** Login Screen
* **Expected Result:** Should display the message “Invalid login parameters.” for incorrect input.
* **Actual Result:** Displays the message “Invalid login parameters.”
* **Remarks:** Pass

When a user enters an incorrect username and password combination, an appropriate error message is displayed.

#### **Test Case 2: New Account Screen (Mandatory Fields)**

* **Test Case ID:** New Account Screen
* **Expected Result:** Should display messages for all required fields if left blank.
* **Actual Result:** Displays messages like “Please enter your name,” “Please enter your phone number,” etc.
* **Remarks:** Pass

When the user tries to submit the form without filling in all mandatory fields, error messages guide the user to complete the form properly.

Test Case 3: New Account Screen (Email Validation)

* **Test Case ID:** New Account Screen
* **Expected Result:** Should display the message “Please enter the correct email.” for invalid email input.
* **Actual Result:** Displays the message “Please enter the correct email.”
* **Remarks:** Pass

When a user enters an invalid email address, an error message prompts the user to correct the email format.

### **Summary**

All the above test cases were executed successfully for this project. The testing phase ensured that all modules worked correctly, validations were properly implemented, and the system met all requirements before deployment.

## ****CHAPTER 11****

## ****IMPLEMENTATION****

### **System Implementation**

System implementation for a **PG (Paying Guest) Renting System** greatly benefits from strong user involvement and active management support. Active participation of end-users (like PG owners, managers, and tenants) in the design and operation of the system has many positive outcomes.

When users are directly involved in the system’s design, they have more opportunities to shape the system as per their daily operational needs and business priorities. This also gives them a sense of ownership and more control over the final outcome, making them more likely to accept and adopt the system smoothly. Incorporating the users’ real-world knowledge and practical experience leads to more accurate and practical solutions.

However, the relationship between users and system developers often faces challenges during implementation. This is commonly known as the **user-designer communication gap**. This gap arises due to differences in roles, work approaches, and terminologies used by users and technical experts.

Below are some examples of typical concerns from both sides for a PG Renting System:

### **User Concerns**

* **Information Needs:** Will the system provide the necessary information for managing PG bookings, room availability, and tenant details?
* **Data Access:** How quickly can data about available rooms, rent payment history, and tenant records be accessed?
* **Data Retrieval:** How easily can I retrieve information on vacant rooms, maintenance schedules, or pending rent payments?
* **Clerical Support:** How much manual effort will be needed to enter tenant details, update room status, and handle booking requests?
* **Daily Operations:** How will the system fit into daily tasks like checking room availability, booking, handling tenant check-in/check-out, and generating rent receipts?

**Designer Concerns**

* **Storage Requirements:** How much storage will be required for master files containing tenant details, room details, booking history, and payment records?
* **Code Complexity:** How many lines of program code are needed to manage room allocation, tenant profiles, payment processing, and generating various reports?
* **CPU Efficiency:** How can the system be optimized to ensure fast processing even when multiple users access it during peak times (e.g., start of a new month when rents are collected)?
* **Data Storage:** What is the most efficient way to store data related to PG rooms, tenants, rent transactions, and complaints or maintenance requests?
* **Database Management:** Which database management system should be used to ensure the system is secure, scalable, and performs efficiently as the number of tenants grows?

### **Bridging the Gap**

Effective system implementation for a PG Renting System requires closing this **user-designer communication gap**. This can be achieved through regular meetings, feedback sessions, and clear documentation.

By fostering collaboration and better understanding between the users (PG owners/managers) and system developers, both user requirements and technical constraints can be addressed properly. This ensures that the system will be reliable, user-friendly, and truly helpful for day-to-day PG management.

As a result, the PG Renting System will successfully streamline operations, improve tenant experience, and support future growth.

## ****CHAPTER 12****

## ****MAINTENANCE****

### **Introduction to Software Maintenance**

In the context of a **PG (Paying Guest) Renting System**, software maintenance refers to any updates or changes made to the system after it has been deployed and delivered to the users. Maintenance is an ongoing process for almost every software product because it is nearly impossible to make any software completely error-free.

Since the input data and user scenarios for a PG Renting System can be very diverse (different tenants, room bookings, payments, etc.), exhaustive testing for every possible situation is impractical. As a result, maintenance activities become essential to fix unexpected issues, enhance existing features, add new functionalities, or adapt to changing business needs and user expectations.

### **Types of Software Maintenance**

Maintenance for a PG Renting System includes various activities to keep the system running smoothly, reliably, and in line with user requirements. The main types of maintenance are described below:

**1. Adaptive Maintenance:**  
Adaptive maintenance involves making changes to the system to meet new requirements or adapt to changes in the environment. For example:

* Integrating with new payment gateways (like UPI, digital wallets).
* Adding support for new features such as online tenant verification or auto-reminders for rent due dates.
* Modifying the system to comply with new rental regulations or data privacy rules.
* Expanding the system to handle multiple PG branches or properties.

**2. Corrective Maintenance:**  
Corrective maintenance focuses on fixing defects or errors found during the operation of the system. This includes:

* Resolving issues like incorrect tenant details, errors in rent calculation, or double booking of rooms.
* Fixing payment processing bugs.
* Correcting glitches in the tenant portal, booking screens, or admin dashboard.
* Addressing any issues reported by PG owners, managers, or tenants during regular usage.

**3. Perfective Maintenance:**  
Perfective maintenance involves improving the system’s performance and usability based on feedback and evolving user needs. For example:

* Enhancing the search feature to find available rooms faster.
* Improving the user interface for easier navigation and a better tenant experience.
* Optimizing backend processes to manage bookings, payments, and complaints more efficiently.
* Adding new reports and dashboards for better PG management insights.

**4. Preventive Maintenance:**  
Preventive maintenance aims to reduce the risk of future system failures by making proactive changes. This includes:

* Applying regular security updates and patches to protect tenant data.
* Optimizing database performance to handle increasing tenant records.
* Monitoring system logs to detect potential issues before they become major problems.
* Performing regular data backups to ensure the safety and availability of all PG-related information.

### **Summary**

By implementing a well-structured approach to software maintenance, the **PG Renting System** can continue to operate reliably and efficiently. Regular maintenance ensures that the system adapts to changing requirements, stays secure, and continues to deliver a smooth experience to PG owners, managers, and tenants.

Effective maintenance ultimately helps maintain user satisfaction, reduces unexpected downtimes, and ensures the PG business runs smoothly and professionally.

**CHAPTER 13**

**Conclusion**

The development of the **“PG Renting System”** marks a significant step forward in the management of **paying guest accommodations**, introducing a reliable, user-friendly, and efficient digital platform that enhances both administrative operations and the tenant experience.

By addressing common challenges faced in traditional PG management — such as manual record-keeping, limited accessibility, inefficient room allocation, delayed rent tracking, and lack of transparency — this system leverages modern technology to optimize resource utilization, streamline processes, and maintain high service standards.

The platform provides a **comprehensive Admin Module** for secure management of room availability, tenant registrations, rent payments, and feedback, along with an **intuitive Tenant/User Module** that simplifies searching for available rooms, booking, rent payment, and profile management.

This project sets a new benchmark for digital solutions in the PG rental sector, demonstrating how convenience, reliability, and smart technology can come together to improve everyday operations. By overcoming the limitations of outdated manual processes, the **PG Renting System** not only saves time and effort for PG owners and managers but also ensures a smooth, transparent, and satisfying experience for tenants.

The successful design and implementation of this project establishes a strong foundation for future enhancements — such as integrating advanced payment methods, automated reminders, and expanding to multiple properties — ensuring continuous growth and improvement in PG rental management.

As the **PG Renting System** continues to evolve, it is well-positioned to lead the way in delivering hassle-free, modern, and secure renting experiences for both owners and tenants, setting a standard for efficiency and customer satisfaction in the PG accommodation industry.