**Project Design Phase-II**

**Solution Requirements (Functional & Non-functional)**

|  |  |
| --- | --- |
| Date | 26-05-2025 |
| Team ID | LTVIP2025TMID21134 |
| Project Name | House Hunt |
| Maximum Marks | 4 Marks |

**Functional Requirements:**

**Solution Requirements**

The House Hunt platform requires both functional and non-functional capabilities to meet user expectations

and ensure system performance.

**Functional Requirements**

Functionally, the system must support user registration and secure authentication for different roles

— renters, property owners, and the admin.

Once logged in, renters should be able to view all listed properties, search using dynamic filters, and

initiate booking requests.

Owners should be able to add, edit, and delete property listings, as well as manage incoming booking requests by approving or rejecting them.

The admin plays a critical role in ensuring the integrity of the platform by approving new property

owners and managing users and listings.

Admins must have access to a dashboard where they can monitor user activities, verify ownership

claims, and handle flagged properties or accounts.

In addition to user-specific features, the system must handle property images, floor plans, virtual tours, and form submissions efficiently.

Real-time status updates for bookings and properties are also part of the essential functional features.

**Non-Functional Requirements**

In terms of non-functional requirements, usability is a top priority. The interface must be clean,

responsive, and intuitive to ensure that users of all types — from tech-savvy owners to first-time

renters — can navigate the platform with ease.

Performance is equally critical; the application must deliver fast response times for actions such as

property searches, booking submissions, and status updates.

Security is fundamental, especially when handling user data and authentication. The platform should implement encrypted password storage, secure login tokens using JWT, and protected routes based on user roles.

It should also provide data validation to prevent injection attacks or other vulnerabilities.

Scalability is important for handling growth. As the number of users and property listings increases,

the backend and database should scale efficiently without affecting speed or uptime. Reliability and

availability must be maintained to ensure minimal downtime and maximum user trust.

Maintainability is achieved by structuring the codebase into reusable modules and following industry- standard development practices. This makes it easier for future developers to understand, extend, or debug the system. Cross-platform compatibility ensures the platform performs consistently across major browsers and devices.

Following are the functional requirements of the proposed solution.

|  |  |  |  |
| --- | --- | --- | --- |
| **NFR-1** | **Usability** | |  | | --- | | The platform should provide a simple, clean UI for all users, including patients and healthcare providers. | |
| **NFR-2** | **Security** | |  | | --- | | Encrypt all data; use role-based access; implement secure login/authentication. | |
| **NFR-3** | **Reliability** | |  |  | | --- | --- | | |  | | --- | | Core features (booking, payments, messaging) must function consistently. | | |
| **NFR-4** | **Performance** | Pages and booking actions should load within 2 seconds; reminders and notifications should be timely. |
| **NFR-5** | **Availability** | |  | | --- | | The system should ensure 99.9% uptime with minimal downtime. | |
| **NFR-6** | **Scalability** | |  | | --- | | Support a growing number of users, providers, and concurrent bookings without degradation. | |