**Project Design Phase-II**

**Data Flow Diagram & User Stories**

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

**Data Flow Diagram (DFD) Overview – House Hunt**

A Data Flow Diagram (DFD) illustrates how data moves through the House Hunt application. It captures the interaction between users (Renters, Owners, Admins) and the system’s core functionalities.

**Level 0 – Context Diagram**

This high-level diagram shows external users interacting with the system:

* **Renters**: Register, log in, browse properties, and book visits.
* **Owners**: Register, get admin approval, list properties, and manage bookings.
* **Admins**: Approve owners, manage users and listings.
* The system communicates with a central **MongoDB** database to store user, property, and booking data.

**Level 1 – Process Breakdown**

Breaks down the core system into modules:

* **User Authentication**: Handles login/signup, JWT generation, and secure storage.
* **Property Management**: Enables property listing updates, image uploads, and saves data.
* **Browsing & Filtering**: Provides search filters and displays matching listings in real time.
* **Booking Management**: Allows booking requests, owner approval, and status updates.
* **Admin Controls**: Admin can manage users, approve owners, and moderate listings.
* **Notifications**: Sends updates and shows real-time booking statuses.

**Data Stores**

* **Users**: Renters, Owners, Admins
* **Properties**: Listings and metadata
* **Bookings**: Requests and status history  
  All data is managed using **MongoDB + Mongoose** for structure and validation.

**External Entities**

* **Renter**: Browses and books properties
* **Owner**: Manages listings and booking responses
* **Admin**: Oversees platform activities  
  Communication occurs via **Express.js REST APIs**.

Let me know if you want this added to a Word or PDF document.



**User Stories**

**User Story Table – DocSpot**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement (Epic)** | **User Story / Task** | **Acceptance Criteria** | **Priority** | **Release** |
| |  | | --- | | **Tenant** |  |  | | --- | |  | | |  | | --- | | Property Discovery & Booking | | As a tenant, I can search for rental properties by location, budget, and features. | Listings returned based on filters and relevance.. | High | Sprint-1 |
|  | Booking Visits | As a tenant, I can book, reschedule, or cancel property visits. | |  | | --- | | Visit status updates in  dashboard  . | | Medium | Sprint-2 |
| |  | | --- | | **Landlord** |  |  | | --- | |  | | |  | | --- | | Listing  Management | | |  |  | | --- | --- | | |  | | --- | | As a landlord, I can create and update  property listings with images and  availability. | | | |  | | --- | | Changes reflect immediately in patient booking system. |  |  | | --- | |  | | High | Sprint-1 |
| **User** | Messaging | As a user, I can chat with the other party in real-time | Messages appear instantly. | Medium | Sprint-2 |
| **Admin** | Profile Management | As a Admin, I can update my profile. | Changes are saved successfully. | High | Sprint-2 |
| **Client** | Payment Processing | As a client, I can make secure payments after job completion. | Payment recorded and shown in transaction history. | High | Sprint-2 |
| **user** | |  | | --- | |  |  |  | | --- | | Review System | | As a user, I can leave a review after project closure. | Review appears on profile. | Medium | Sprint-2 |
| **Admin** | User Moderation | As an admin, I can review reported users and take action. | Reports and actions logged. | High | Sprint-1 |