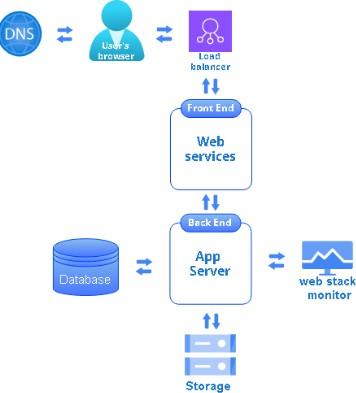
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 27 June 2025 |
| Team ID | LTVIP2025TMID20380 |
| Project Name | HouseHunt : Finding your perfect rental Home |
| Maximum Marks | 4 Marks |

**Technical Architecture – HouseHunt**

HouseHunt is built using a **client-server architecture**, ensuring smooth interaction between users, agents, and admins. The system is divided into three main layers: **Frontend**, **Backend**, and **Database**. RESTful APIs connect the layers, enabling secure and efficient data exchange. Real-time chat and notifications are supported using **Socket.IO**.



**Architecture Guidelines – HouseHunt**

* The system includes core blocks:

1. **Frontend:** React.js (Material UI, Bootstrap)
2. **Backend:** Node.js + Express.js (REST APIs)
3. **Database:** MongoDB (User, Complaint, Chat, Feedback data)

* **Infrastructure:**
  1. Local setup for development
  2. Cloud deployment via Vercel (frontend), Render or Railway (backend), MongoDB Atlas (database)
* **External Interfaces:**
  1. Gmail SMTP for emails
  2. Google OAuth for login
  3. Optional: Twilio for SMS
* **Data Storage:**
  1. All structured data in MongoDB
  2. Files/images stored via Firebase or AWS S3 (optional)
* **ML Model (Optional):**

1. Future-ready for smart routing or auto-prioritization using ML

**Table-1: Components & Technologies**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | **User Interface** | How user interacts with the application (Web UI, etc.) | **React.js**, **HTML**, **CSS**, **JavaScript**, **Material UI**, Bootstrap |
| 2 | **Backend** | Server-side logic, API routes, authentication, CRUD operations | **Node.js**, **Express.js** |
| 3 | **Authentication** | Handles user login, registration, and JWT-based session contro | **JSON Web Token (JWT)** |
| 4 | **API Testing** | Testing API endpoints manually during development | **Postman** |
| 5 | **Database** | Data storage for users, complaints, chats, etc. | **MongoDB (NoSQL)** |
| 6 | **State Management** | Handling local component state and API response handling | **React useState, useEffect, Axios** |
| 7 | **Image Upload** | Uploading and saving property images to server | **Multer (Node middleware)** |
| 8 | **Routing** | Navigating between pages in frontend | **React Router DOM** |
| 9 | **Admin Panel** | Admin functionalities like owner approval | **Custom-built React Components** |
| 10 | **Infrastructure** | Hosting backend/frontend on cloud/local | **Localhost, Render, Railway, Cloud Foundry, Kubernetes** |

**Table-2: Application Characteristics**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology Used** |
| 1 | **Open-Source Frameworks** | Libraries and frameworks used | **React.js**, **Express.js**, **Node.js**, **Mongoose**, **Socket.io** |
| 2 | **Security Implementations** | Authentication, Authorization, Data Protection | **JWT**, **bcrypt.js**, **CORS**, **HTTPS**, **SHA-256**, **Helmet** |
| 3 | **Scalable Architecture** | Modular design for performance and growth | **3-tier architecture**, **Microservices-ready**, **REST APIs** |
| 4 | **Availability** | Ensures uptime, handles traffic | **Load Balancers**, **Cloud Deployment**, **Clustered MongoDB** |
| 5 | **Performance** | Optimized code for response time and user experience | **Axios**, **CDN**, **Caching (Redis optional)**, **Lazy Loading** |

**References**

1. C4 Model for Visualising Software Architecture. Retrieved from: <https://c4model.com/>
2. IBM Developer: Online Order Processing System During Pandemic. Retrieved from: [https://developer.ibm.com/patterns/online-order-](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/) [processing-system-during-pandemic/](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)
3. IBM Cloud Architecture Center. Retrieved from: <https://www.ibm.com/cloud/architecture>
4. AWS Architecture Center. Retrieved from: <https://aws.amazon.com/architecture>
5. “How to Draw Useful Technical Architecture Diagrams” – Medium Article. Retrieved from: [https://medium.com/the-internal-startup/how-to-](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d) [draw-useful-technical-architecture-diagrams-2d20c9fda90d](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)