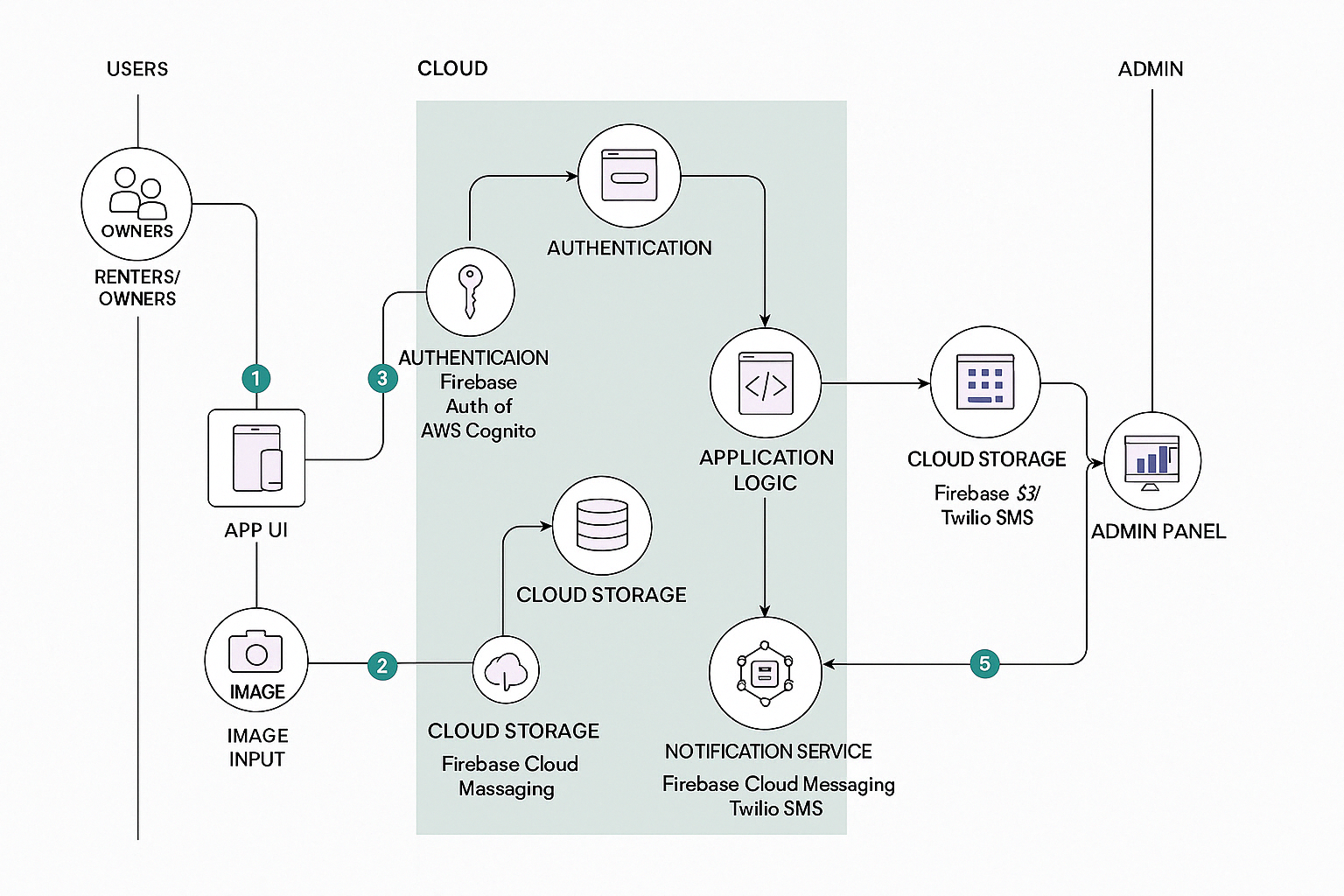
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 13 April 2025 |
| Team ID | SWTID1743517551 |
| Project Name | HOUSEHUNT – A HOUSE RENT APP |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | User Interface | How users browse listings, filters, book houses | HTML, CSS, JavaScript, React.js |
| 2 | Application Logic-1 | Handles user authentication, session, and role management | Node.js / Express.js |
| 3 | Application Logic-2 | Listing management, booking logic, payment flow | Node.js / Express.js |
| 4 | Database | Stores user data, listings, reviews, payment logs | MongoDB |
| 5 | Cloud Database | Scalable cloud-based storage for listings and transactions | Firebase / Amazon RDS |
| 6 | File Storage | Stores images of rental houses and user-uploaded documents | Firebase Storage / Local Filesystem |
| 7 | External API-1 | Address autocomplete, map embedding, rental price suggestions | Google Maps API / OpenStreetMap / RentCast |
| 8 | Infrastructure (Server/Cloud) | Hosting, scaling, and deployment of the app | Vercel / Heroku / AWS / Render |
| 9 | Notifications | Sends email/SMS/Push alerts for booking updates and reminders | Firebase Cloud Messaging / Twilio / SendGrid |
| 10 | Analytics | Tracks user behavior, engagement, and app performance | Google Analytics / Mixpanel |
| 11 | Admin Dashboard | Manages listings, users, payments, reviews and issues | React.js / Node.js / MongoDB |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | The app utilizes open-source tools for development, enabling transparency, community support, and cost-efficiency. | React Native (Frontend), Node.js (Backend) |
|  | Security Implementations | App includes role-based access, secure login with JWT, HTTPS encryption, and server-side input validation to prevent data breaches. | JWT, HTTPS, Bcrypt, OWASP Secure Headers |
|  | Scalable Architecture | The architecture follows a modular microservices pattern with decoupled front-end and back-end, allowing horizontal scaling when needed. | Docker, Kubernetes, RESTful APIs |
|  | Availability | Load balancers and cloud-hosted backend ensure high availability and minimal downtime for users searching for or listing rental properties. | NGINX, AWS Elastic Load Balancer |
|  | Performance | Optimized for fast response time using caching, asynchronous APIs, and content delivery networks (CDNs) for static assets. | Redis Cache, Cloudflare CDN, Lazy Loading in React |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)