INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

System Requirements Specification

Course Title : Software Engineering Lab

Course Code ; CSE-3638

Submitted by

- Nahian Subah Ishma_C223286
- Rehnuma Tasneem_C223288
- Saima Kawsar_C223297
- Sakaratul Ara Tasmia_C223298
- Tahsin Islam Nafisa_C223311

Semester : 6th Section : 6CF

Submitted to

Sultana Tasnim Jahan
Assistant Lecturer
Dept. of CSE , IIUC

Table of Contents

1.Preface

1.1 Expected Readership

This document is written for all stakeholders involved in the project. This includes:

- Project Managers – to keep track of the work,
- Developers – to understand what to build,
- Testers – to test the system properly,
- System Designers/Architects – to design the system structure,
- Users or Reviewers – to understand what the platform will do.

1.2 Version History

Version	Date	Why Updated	What Was Changed
1.0	2025-05-18	First version of the document	Full system requirements and features added
1.1	2025-05-21	Added system architecture part	Explained 3-layer structure and data flow

1.3 Why This New Version Was Made

This new version (v1.1) was made to improve the document. It now includes a better explanation of how the system works behind the scenes using a 3-layer architecture. This makes the platform more organized and easier to develop, test, and maintain.

1.4 Summary of Changes in Each Version

- ✓ Version1.0:
  – Wrote the basic system requirements.
  – Explained what the system will do and who will use it.
- ✓ Version1.1:
  –Added clear explanation of the 3-layer system (frontend, backend, database).
  –Updated the structure to show how each part of the system works.
  –Made the document easier to understand for everyone.

2. Introduction

2.1 Purpose of the System

The purpose of this Real Estate Platform is to make it easier for buyers, agents, and admins to connect in one place for buying and selling properties. It aims to remove the hassle of offline communication and make the whole process smoother, faster, and more reliable. The platform ensures an easier experience in listing, managing, and purchasing properties.

2.2 System Functions

The system includes several useful features to help users:

- Login system with different roles (User, Agent, Admin)
- Property listing, editing, reviewing, and buying
- Users can add properties to their wishlist and send offers
- Secure payment system using Stripe
- Admin can manage users, agents, properties, and reviews
- Option to handle property advertisements

2.3 Interactions with Other Systems

The platform connects with some external tools and technologies:

- **Stripe** – used for secure online payments
- **Firebase** – used for user authentication and real-time updates
- **MongoDB** – the database to store user, property, and transaction data
- **Node.js + Express** – backend handles logic and APIs
- **React** – frontend communicates with the backend to show data and take actions

2.4 Business and Strategic Fit

This platform supports the growth of the online real estate market by providing a user-friendly, secure, and organized solution for property transactions. It helps businesses and users save time, build trust, and operate more smoothly. Also, the system is scalable — meaning it can grow with more features or users in the future, supporting long-term goals and business success.

3. Glossary

Term	Meaning
Agent	A user who can add and manage property listings.
Admin	A user who controls the whole platform and manages users and content.
Fraudulent Agent	An agent marked as untrustworthy by the admin.
Wishlist	A list where users save properties they like.
Stripe	A service used to handle safe online payments.
JWT (JSON Web Token)	A small digital key used to keep users securely logged in.
API	A tool that helps the frontend and backend talk to each other.
CRUD	Basic actions like creating, reading, updating, and deleting data.
RBAC (Role-Based Access Control)	A system that gives users different permissions based on their roles (user, agent, admin).
Authentication	The process of checking who the user is (like login).
Authorization	Deciding what a logged-in user is allowed to do.
Firebase	A Google tool used to manage login, real-time updates, and storing user data.
Frontend	The part of the website that users see and interact with.
Backend	The part of the system that works behind the scenes to process data and handle logic.
MongoDB	A database where all the information (users, properties, etc.) is stored.
React	A JavaScript library used to build the user interface (UI) of the website.
Node.js	A tool that lets JavaScript run on the server (backend).
Express.js	A framework used with Node.js to make backend APIs.

4.User-Requirements-Definition

4.1 Functional Requirements

1.Authentication & Authorization

1a. System shall allow email/password login and social login.

1b. System shall support role-based route access.

2. Role Management

2a. System shall support three roles: User, Agent, Admin.

2b. Admin shall be able to change roles or mark agents as fraud.

3. Wishlist & Offers

3a. Users can add or remove properties from wishlist.

3b. System shall validate price range for offers.

3c. Agent shall approve/reject offers.

4. Payment Processing

4a. Users shall be able to pay via Stripe for approved offers.

4b. System shall update status to “Paid” and store transaction ID.

5. Property Management

5a. Agents can add properties (default status: Pending).

5b. Admin can approve/reject properties.

5c. Fraud agents’ properties shall not appear on homepage.

6.Reviews & Reports

6a. Users can submit/delete their reviews.

6b. Admin can delete any review.

6c. Users can report properties; Admin will manage reports.

7.Advertisement

7a. Admin can add/remove advertisements for properties.

7b. Advertised properties shall appear on homepage.

4.2 Non-Functional Requirements

1. Security

1a. Secure Endpoints with Encrypted JWT Tokens

1b. Ensure Safe and Secure Stripe Payment Processing

2. Performance

2a. Efficient MongoDB Queries

2b. Exclude Properties Listed by Fraudulent Agents Using Aggregation Pipeline

3. Usability

3a. User-Friendly Interface with Clear Navigation

3b. Responsive Design for Various Devices

4. Maintainability

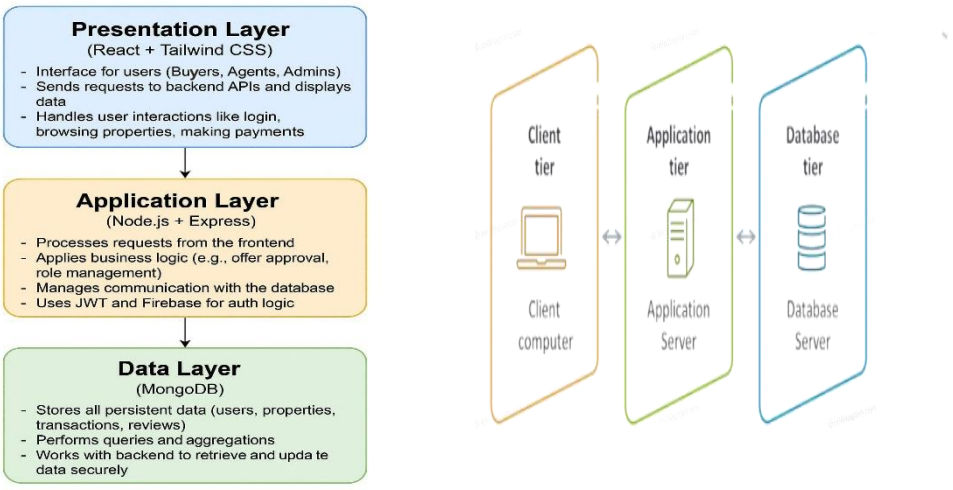
4a. Modular React Components for Frontend

4b. Modular Backend Architecture for Easy Updates and Scalability

5. System Architecture

This Real Estate Platform is built using a **3-layer architecture**, which means the whole system is divided into three main parts. Each part has its own important job to do. This way, the system becomes easier to build, understand, and improve over time. The three layers work together smoothly to provide a fast, secure, and user-friendly experience for everyone—whether they are buyers, agents, or admins.

1. **Presentation Layer (Frontend/UI)**
   - ✓ **Technology Used:** React + Tailwind CSS
   - ✓ **Role:**
     - This is what the users see and use — Buyers, Agents, and Admins.
     - It sends requests to the backend and shows the data that comes back.
     - Handles things users do like logging in, searching properties, and paying.
2. **Application Layer (Backend/Logic)**
   - ✓ **Technology Used:** Node.js + Express
   - ✓ **Role:**
     - Takes requests from the frontend and processes them.
     - Applies the rules of the platform, like approving offers and managing user roles.
     - Talks with the database to get or save information.
     - Uses JWT and Firebase to keep the system secure with login and permissions.
3. **Data Layer (Database)**
   - ✓ **Technology Used:** MongoDB
   - ✓ **Role:**
     - Saves all important information like users, properties, payments, and reviews.
     - Runs queries to find and organize data.
     - Works with the backend to safely give and update data when needed.



3-Layer Architecture

6.System-Requirements-Specification

6.1 Functional Requirements (Detailed)

1. Authentication & Roles

1a. Users should be able to register/login using email or social login (Google/Facebook).

1b. System should support three roles: Regular User, Agent, and Admin.

2. Property Exploration & Actions

2a. Users can view all verified properties.

2b. Users can add properties to their wishlist.

2c. Users can make offers within a given price range.

2d. Users can remove properties from their wishlist at any time.

2e. Users should be able to see the status of their offers: pending, approved, rejected, or paid.

3. Reviews & Feedback

3a. Users can submit reviews on properties they've interacted with.

3b. Users can view and delete their own reviews from their dashboard.

4. Payments

4a. If an offer is approved, users can make payments via Stripe.

4b. Paid properties should appear in the Bought Properties section.

5. Agent Features

5a. Agents can add new properties to the platform.

5b. Agents can track both requested and sold properties.

5c. Agents can approve or reject user offers.

6. Admin Features

6a. Admins can approve or reject properties submitted by agents.

6b. Admins can manage user accounts and promote or demote user roles.

6c. Admins can mark agents as fraud and hide their properties.

6d. Admins can delete any user review.

6e. Admins can manage reported properties.

7. Usability & Access

7a. All users should have dashboards based on their assigned roles.

7b. The interface should be fully responsive and mobile-friendly.

7c. The system should be accessible 24/7 with minimal downtime.

6.2 Non-Functional Requirements (Detailed)

1.Performance

1a. The system must handle at least 1000 concurrent users without significant performance degradation.

1b. Average page response time should be under 2 seconds.

2.Security

2a. All passwords must be stored using encryption.

2b. Role-based access control must be enforced for all routes and actions.

2c. All payment transactions must be securely processed using Stripe.

3.Scalability

3a. The system architecture should allow dynamic addition of users and properties without major refactoring.

3b. The system should be deployable on scalable cloud infrastructure (e.g., AWS, GCP).

4.Availability

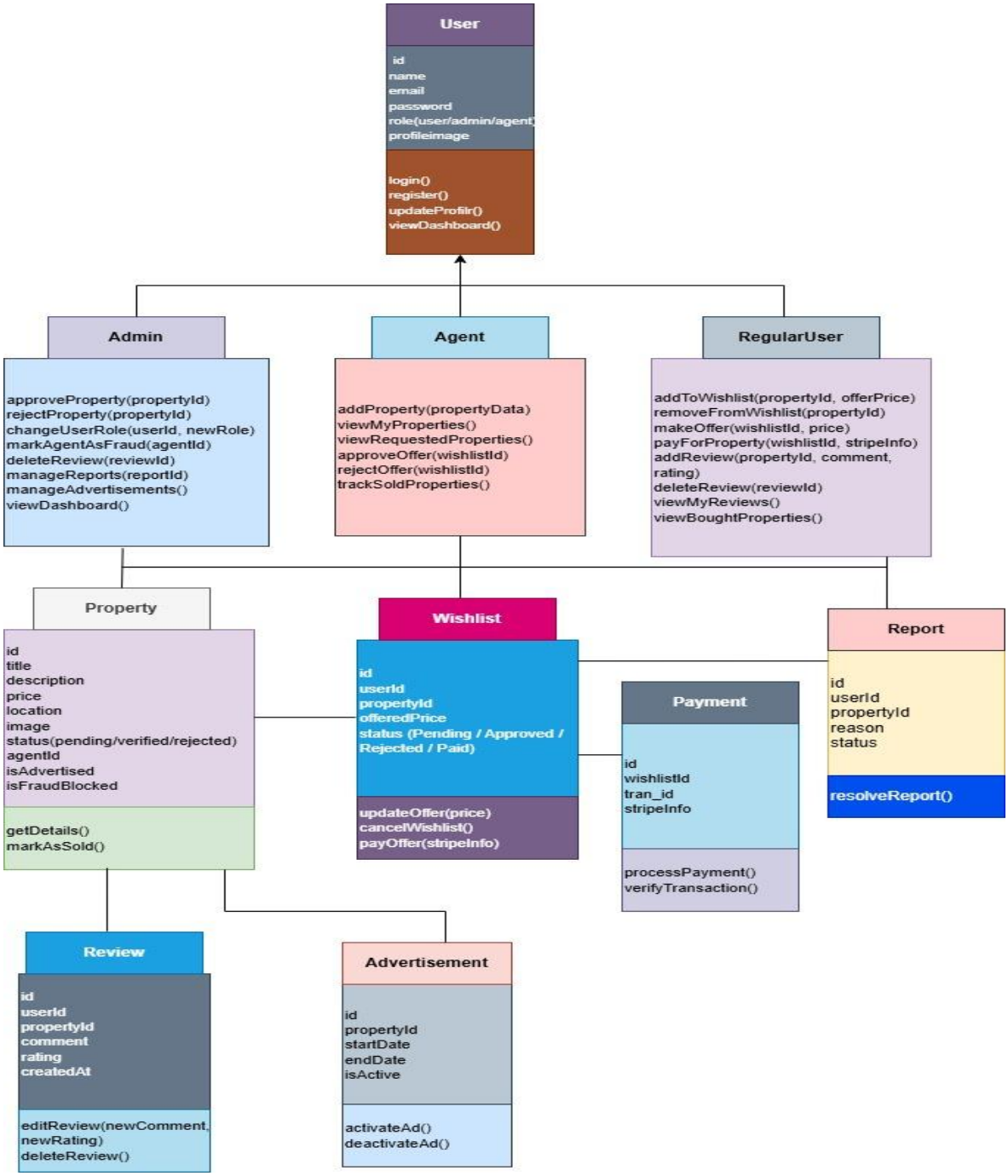
4a. The system should maintain 99% uptime monthly.

4b. Scheduled maintenance should be communicated to users at least 24 hours in advance.

5.Usability

5a. The interface must be fully responsive for mobile, tablet, and desktop

7. System Model



Object Model

8. System Evolution

8.1 Fundamental Assumptions

- Different types of users will use the platform, including regular users who want to buy or rent properties, agents who add and manage property listings, and admins who control the system.
- Users will access the platform through web browsers on devices like smartphones, tablets, and desktop computers.
- The system uses third-party services such as Stripe for payments and Google/Facebook for social login.
- The platform expects stable internet connections and servers to be available most of the time.
- Properties added by agents must be checked and approved by admins before showing publicly to ensure trust.

8.2 Anticipated Changes

- The platform will need to handle more users, agents, and property listings as it grows, without slowing down.
- As hardware and technology improve, the system should be easy to upgrade to faster servers and newer devices.
- Users may want new features like better property search options or new payment methods, so the system must be flexible to add these without big changes.
- Security rules and payment methods may change over time, so the platform must be ready to update accordingly.
- More people will use mobile devices, so the platform's design should improve to work well on phones and tablets.
- The system will be built in parts (modular), so updates and fixes can be made easily without breaking the whole platform.
- In the future, the platform might add new payment options, AI suggestions for properties, and new user roles for more features.

9.Appendices

9.1 Hardware Requirements

- Minimum: 4GB RAM, 2 CPUs, 20GB SSD for backend server
- Recommended: Cloud servers that can grow

9.2 Database Description

- MongoDB collections: users, properties, reviews, transactions, ads
- Relations handled with references and aggregation queries

10. Index

This section provides the Alphabetic Index, Index of Diagrams, and Index of Functions mentioned in the SRS document.

10.1 Alphabetic Index

10.2 Index of Diagrams

10.3 Index of Functions