Software Requirements Specification
SRS

cohabit

Gruppe: Albin Qerkinaj, Veljko Stojadinovic, Besim Kurteshi, Berkant-Yasin Sener

Dokument-Historie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
0.01	Draft	2025-11-15	Albin Qerkinaj	Initial
1.00	Initial Version	2025-12-07	Albin Qerkinaj	Finished doc

Inhaltsverzeichnis

1. Einleitung

1.1. Zweck

Das System dient als Plattform, die es Nutzern ermöglicht, Wohnungsanzeigen auf Basis von authentischen Bewertungen und detaillierten Beschreibungen transparent einzuschätzen. Dies soll Miet- und Kaufentscheidungen durch zusätzliche Informationen (Nachbarschaft, Lärmpegel etc.) unterstützen.

1.1.1. Hintergründe und Ziele des Projekts

Aktuelle Immobilienanzeigen bieten oft nur ein geschöntes Bild einer Wohnung. Informationen über soziale Faktoren (Nachbarn) oder Umweltfaktoren (Lärm, Lichtverhältnisse) fehlen oft. Ziel: Das Ziel des Projekts ist es, maximale Transparenz auf dem Wohnungsmarkt zu schaffen, indem subjektive Erfahrungen von Vormietern gesammelt und strukturiert dargestellt werden.

1.1.2. organisatorische Einbettung

Das Projekt wird im Rahmen des Moduls Software Engineering an der Hochschule Campus Wien als Semesterprojekt durchgeführt.

1.1.3. technische, wirtschaftliche, organisatorische, ergonomische Ziele

- Technische Ziele: Einsatz moderner Web-Technologien für eine responsive Anwendung; hohe Skalierbarkeit.
- Wirtschaftliche Ziele: Langfristige Finanzierung durch Werbeeinnahmen.
- Organisatorische Ziele: Agile Entwicklung nach Scrum/Kanban.
- Ergonomische Ziele: Intuitive Benutzeroberfläche (Usability), Barrierefreiheit und optimierte Darstellung auf mobilen Endgeräten (Mobile First).

1.2. Produktumfang

Das System **cohabit** ist eine webbasierte Plattform, die Transparenz auf dem privaten Wohnungsmarkt schafft. Der Fokus liegt auf der Erfassung und Darstellung authentischer Erfahrungsberichte von Vormietern und aktuellen Mietern.

Der Funktionsumfang gliedert sich in folgende Kernbereiche:

- **Benutzerverwaltung:** Das System unterscheidet zwischen Mietinteressenten (User) und Immobilienverwaltern (Anbieter). Es ermöglicht die Registrierung und sichere Authentifizierung (Login) beider Benutzergruppen.
- **Wohnungsverwaltung:** Anbieter können digitale Profile für Wohneinheiten erstellen, bearbeiten und verwalten. Diese Profile dienen als Basis für Bewertungen.
- **Bewertungssystem:** Registrierte Nutzer können Wohnungen anhand definierter Kriterien (z. B. Lärm, Helligkeit) bewerten und Kommentare hinterlassen.
- **Recherche & Information:** Eine Suchfunktion ermöglicht es Nutzern (auch ohne Registrierung), gezielt nach Wohnungen und deren Bewertungen zu suchen.

- **Qualitätssicherung:** Mechanismen zur Validierung von Eingaben und zur Wahrung der Anonymität (keine Klarnamen in Bewertungen) sind integriert.

Detaillierte funktionale Anforderungen (Muss-, Soll-, Kann-Kriterien) sind in Kapitel 3 definiert. Die technischen Rahmenbedingungen und Qualitätsmerkmale finden sich in Kapitel 4 (Nicht-funktionale Anforderungen).

1.3. Definitionen, Akronyme, Abkürzungen

- **System:** Bezeichnet die entwickelte Webanwendung (Softwarelösung) inklusive aller Hintergrundkomponenten (Datenbanken, Schnittstellen, Frontend). In den Anforderungen tritt das System als der Akteur auf, der Funktionen bereitstellt („Das System muss...“).
- **Wohnungsanzeige:** Bezeichnet den Datensatz innerhalb des Systems, der eine spezifische Wohneinheit repräsentiert. Sie dient als Ankerpunkt für Bewertungen, Detailinformationen (Adresse) und Statistiken. *Abgrenzung:* Es handelt sich technisch um ein Profil zum Zweck der Bewertung, **nicht** um ein rechtsverbindliches Inserat zur Vermittlung, Vermietung oder zum Verkauf.
- **User (Allgemein):** Sammelbegriff für die Benutzergruppe der Mieter oder Mietinteressenten (im Gegensatz zum „Anbieter“). In den Anforderungen wird der Begriff „User“ verwendet, wenn eine Funktion für **alle** Nutzergruppen (registriert und unregistriert) gilt (z. B. Suchen).
    - **Unregistrierter User (Gast):** Ein User ohne Anmeldung. Er hat nur lesenden Zugriff (Suchen, Details ansehen).
    - **Registrierter User:** Ein User mit authentifizierten Account. Er besitzt schreibende Rechte (Bewerten, Kommentieren, Merkliste).
- **Anbieter:** Bezeichnet die Benutzerrolle für die Verwaltungsebene einer Immobilie (z. B. private Vermieter, Hausverwaltungen, Genossenschaften). Ein Anbieter besitzt privilegierte Rechte, um Wohnungsanzeigen zu erstellen, zu bearbeiten, zu deaktivieren und deren Statistiken (Aufrufe) einzusehen.

2. Allgemeine Übersicht

2.1. Beschreibung der Ausgangssituation (Ist-Zustand)

Derzeit ist der Wohnungsmarkt von Intransparenz geprägt. Mietinteressenten müssen sich auf die oft beschönigten Angaben in Immobilieninseraten verlassen. Informationen über soziale Faktoren (Nachbarschaft, Lärm) oder Umweltbedingungen (Lichtverhältnisse, Hellhörigkeit) sind vor Vertragsabschluss kaum zugänglich. Es existiert keine zentrale Plattform, die subjektive Erfahrungen von Vormietern strukturiert sammelt und bereitstellt.

2.2. Produkteinsatz

2.2.1. Anwendungsbereiche

Das System wird als Webanwendung für den privaten Wohnungsmarkt genutzt. Es dient der Informationsbeschaffung vor Mietentscheidungen sowie der Dokumentation von Wohnqualität durch Mieter.

2.2.2. Zielgruppen, Qualifikationsniveau

- **User (Mieter):** Personen auf Wohnungssuche oder aktuelle Mieter. Technisches Vorwissen: Gering bis Durchschnitt.
- **Anbieter (Vermieter):** Private Vermieter, Hausverwaltungen. Technisches Vorwissen: Heterogen.

2.2.3. Betriebsbedingungen

Nutzung über gängige Webbrowser (Chrome, Firefox, Safari, Edge) auf Desktop-PCs, Tablets und Smartphones. Das System ist 24/7 verfügbar.

2.2.4. allgemeine Restriktionen

Einhaltung der DSGVO (Datenschutzgrundverordnung) ist zwingend, insbesondere da Adressen und Meinungen verarbeitet werden. Keine Speicherung von Klarnamen der Nachbarn oder Vermieter in den öffentlichen Texten.

2.2.5. Annahmen und Abhängigkeiten

- **Internetverbindung:** Es wird angenommen, dass die Endgeräte der Benutzer über eine stabile Breitband-Internetverbindung verfügen, da keine Offline-Funktionalität („Offline First“) vorgesehen ist.
- **Browser-Kompatibilität:** Es wird davon ausgegangen, dass die Benutzer moderne Webbrowser nutzen, die JavaScript und CSS Grid unterstützen. Legacy-Browser (z. B. Internet Explorer) werden nicht unterstützt.

2.3. Produktumfeld

2.3.1. Systemschnittstelle

Die Systemschnittstellen beschreiben die internen und externen Verbindungspunkte der Softwarekomponenten. Da das System als verteilte Webanwendung konzipiert ist, stehen hier die API und die Datenbankanbindung im Fokus.

- **API-Schnittstelle (Backend zu Frontend):** Das Backend (Quarkus) stellt eine RESTful API zur Verfügung, über die das Frontend (Svelte) kommuniziert.
    - **Protokoll:** HTTPS
    - **Datenformat:** JSON (JavaScript Object Notation)
    - **Authentifizierung:** Token-basiert (z.B. JWT - JSON Web Token) für geschützte Endpunkte (Bewertung abgeben, Wohnungsanzeige anlegen).
- **Datenbank-Schnittstelle:** Das Backend verbindet sich über JDBC oder Reactive SQL Clients mit der PostgreSQL-Datenbank. Hier werden relationale Daten (User, Wohnungsanzeigen, Bewertungen) persistent gespeichert.

2.3.2. Benutzerschnittstelle

Die Benutzerschnittstelle (Graphical User Interface - GUI) ist der zentrale Kontaktpunkt für die Akteure „User“ und „Anbieter“. Sie wird als moderne Single-Page-Application (SPA) mit Svelte realisiert.

- **Designprinzipien**:
  - Responsive Web Design: Die Oberfläche passt sich automatisch an verschiedene Bildschirmgrößen an (Desktop, Tablet, Smartphone). Dies wird durch CSS Media Queries und ein flexibles Grid-System gewährleistet.
  - Mobile First: Die Bedienung und Darstellung werden primär für mobile Endgeräte optimiert, um eine Nutzung vor Ort (z.B. bei einer Wohnungsbesichtigung) zu ermöglichen.
- **Struktur der GUI:**
  - Navigation: Eine globale Navigationsleiste ermöglicht jederzeit Zugriff auf die Kernfunktionen: „Suche“, „Anmelden/Registrieren“ und „Profil“.
  - Darstellung von Wohnungsanzeigen: Suchergebnisse werden als übersichtliche „Cards“ (Kacheln) dargestellt, die ein Vorschaubild und die Gesamtbewertung (Sterne-Rating) enthalten.
  - Eingabemasken: Formulare (z.B. für das Erstellen einer Bewertung oder Wohnungsanzeige) verfügen über direkte Validierung (z.B. „Pflichtfeld fehlt“), um Fehleingaben zu minimieren.
- Sprache: Die gesamte Benutzeroberfläche ist in deutscher Sprache gehalten.

2.3.3. Hardwareschnittstelle

Da es sich um eine Webanwendung handelt, gibt es keine direkten physischen Schnittstellen zu Sensoren oder Aktoren.

- **Clientseitig**: Das System muss auf handelsüblichen Endgeräten (Smartphones, Tablets, Desktop-PCs, Laptops) lauffähig sein, die über eine Internetverbindung verfügen.
- **Serverseitig**: Das System läuft in einer containerisierten Umgebung, die auf Standard-Serverhardware gehostet wird.

2.3.4. Softwareschnittstelle

Das System interagiert mit folgenden Softwarekomponenten und Technologien:

- **Betriebssystem:** Serverseitige Ausführung auf Linux-basierten Systemen (Container-Umgebung).
- **Datenbank:** PostgreSQL zur persistenten Speicherung von Stamm- und Bewegungsdaten.
- **Backend-Frameworks:** Quarkus (Java) für die Bereitstellung der REST-API und der Geschäftslogik.
- **Frontend-Framework:** Svelte zur Darstellung der Benutzeroberfläche im Browser.
- **Browser:** Unterstützung aller modernen Webbrowser (Chrome, Firefox, Safari, Edge) in aktuellen Versionen.

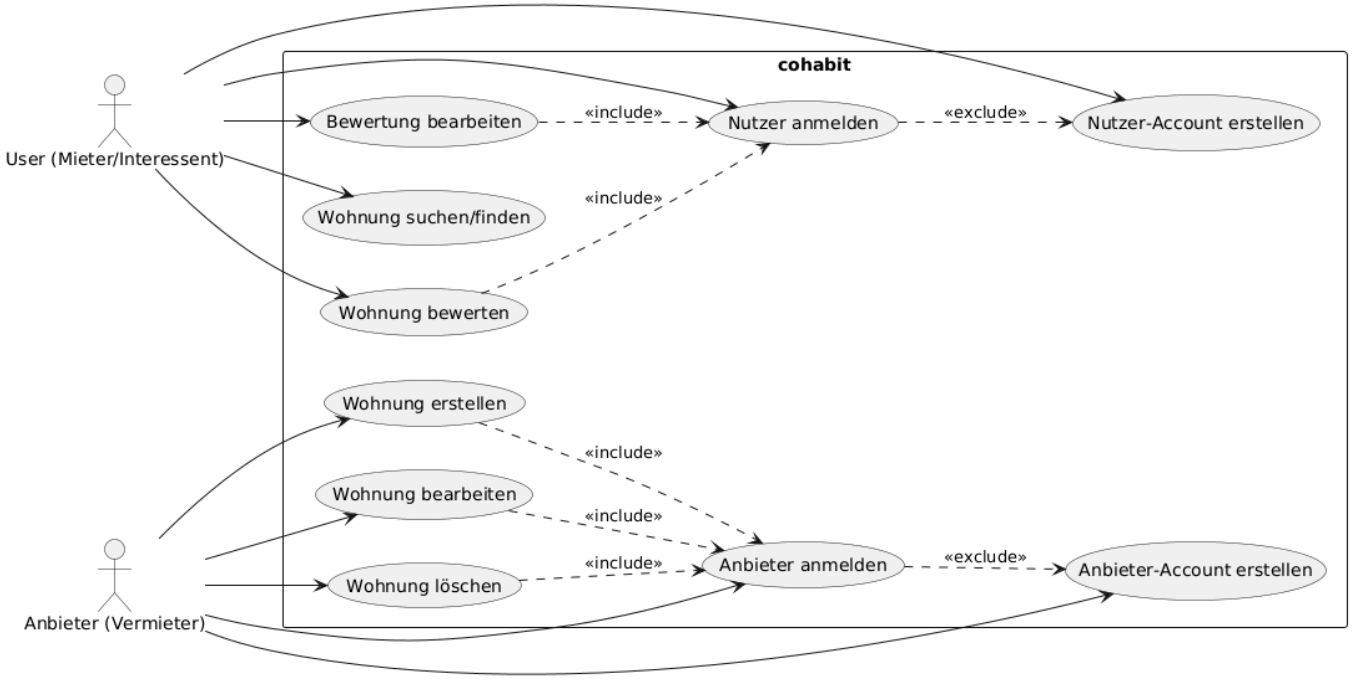
2.3.5. Kommunikationsschnittstelle

- **Protokolle:** Die Kommunikation zwischen Client (Svelte) und Server (Quarkus) erfolgt über HTTPS unter Verwendung von RESTful APIs
- **Datenformat:** Der Datenaustausch erfolgt im JSON-Format.
- **Datenbankverbindung:** Das Backend kommuniziert über TCP/IP (JDBC/pgx) mit der PostgreSQL-Datenbank.

2.3.6. Speicherbeschränkungen

- **Uploads:** Begrenzung der Upload-Größe für Bilder auf maximal 5 MB pro Datei, um die Serverlast und Speicherkosten zu minimieren.
- **Datenbank:** Textfelder für Bewertungen und Beschreibungen werden auf eine vernünftige Zeichenanzahl begrenzt (z.B. max. 2000 Zeichen pro Bewertung), um Missbrauch zu verhindern.

2.4. Produktfunktionalität



Feld	Inhalt
Use Case	Wohnungsanzeige suchen
Akteur	User (auch unregistriert)
Beschreibung	Der Akteur sucht anhand einer Adresse oder Kriterien nach einer Wohnungsanzeige.
Auslöser (Stimulus)	User gibt Suchbegriff in das Suchfeld ein.
Reaktion (Response)	Das System zeigt eine Liste von Wohnungsanzeigen an, die den Kriterien entsprechen.

Feld	Inhalt
Comment	Keine Authentifizierung erforderlich (für alle zugänglich).

Feld	Inhalt
Use Case	Wohnungsanzeige bewerten
Akteur	Registrierter User
Beschreibung	Der User gibt eine Sterne-Bewertung und einen optionalen Textkommentar zu einer Wohnungsanzeige ab.
Auslöser (Stimulus)	User klickt auf "Bewerten" auf einer Wohnungsanzeige.
Reaktion (Response)	1. System prüft, ob User eingeloggt ist. 2. System zeigt Bewertungsformular an. 3. User füllt Formular aus und sendet ab. 4. System validiert Eingabe (keine Klarnamen). 5. System speichert Bewertung und aktualisiert den Durchschnitt. 6. System zeigt Erfolgsmeldung an.
Comment	Vorbedingung: User ist eingeloggt; Wohnungsanzeige existiert im System.

Feld	Inhalt
Use Case	Bewertung bearbeiten
Akteur	Registrierter User
Beschreibung	Der User ändert die Daten einer bestehenden Bewertung.
Auslöser (Stimulus)	User wählt eine seiner Bewertungen und klickt „Bearbeiten“.
Reaktion (Response)	1. System lädt die aktuellen Daten der Bewertungen. 2. User ändert Bewertungen. 3. System validiert Änderungen. 4. System aktualisiert den Datensatz persistent. 5. System bestätigt die Änderung.
Comment	Vorbedingung: User ist eingeloggt und Ersteller der Bewertung.

Feld	Inhalt
Use Case	Bewertung löschen
Akteur	Registrierter User
Beschreibung	Der User entfernt eine Bewertung dauerhaft.
Auslöser (Stimulus)	User klickt bei einer Bewertung auf „Löschen“.
Reaktion (Response)	1. System fordert Bestätigung an („Sind Sie sicher?“). 2. User bestätigt. 3. System löscht den Eintrag aus der Datenbank (oder setzt Status auf inaktiv). 4. System aktualisiert die Listenansicht.
Comment	Vorbedingung: User Ist eingeloggt und ist Ersteller der Bewertung.

Feld	Inhalt
Use Case	Anbieter-Account erstellen (Registrieren)
Akteur	Anbieter (Vermieter, Hausverwaltung)
Beschreibung	Der Akteur registriert sich im System, um Wohnungsanzeigen verwalten zu können.
Auslöser (Stimulus)	Akteur klickt auf „Als Anbieter registrieren“.
Reaktion (Response)	1. System zeigt Registrierungsformular an. 2. Akteur gibt Daten ein (E-Mail, Passwort etc.). 3. System validiert Eingaben (Passwort-Stärke, E-Mail-Format). 4. System legt Account an und speichert Passwort als Hash. 5. System bestätigt die Registrierung.
Comment	Vorbedingung: Akteur ist nicht eingeloggt.

Feld	Inhalt
Use Case	Anbieter anmelden (Login)
Akteur	Anbieter
Beschreibung	Der Akteur authentifiziert sich gegenüber dem System.
Auslöser (Stimulus)	Akteur gibt Zugangsdaten ein und klickt „Login“.
Reaktion (Response)	1. System prüft Anmeldedaten gegen die Datenbank. 2. Bei Erfolg: System erstellt JWT-Token und leitet zum Dashboard weiter. 3. Bei Misserfolg: System zeigt Fehlermeldung.
Comment	Vorbedingung: Account muss bereits existieren.

Feld	Inhalt
Use Case	Wohnungsanzeige erstellen
Akteur	Anbieter
Beschreibung	Der Anbieter legt eine neue Wohnungsanzeige mit Details und Bildern an.
Auslöser (Stimulus)	Anbieter klickt im Dashboard auf „Neue Wohnungsanzeige hinzufügen“.
Reaktion (Response)	1. System prüft Berechtigung (JWT). 2. System zeigt Eingabemaske an. 3. Anbieter füllt Pflichtfelder aus und lädt Bilder hoch. 4. System prüft Limits (Bild max. 5 MB). 5. System speichert die Anzeige in der Datenbank. 6. System zeigt die neue Anzeige in der Übersicht.
Comment	Vorbedingung: Anbieter ist eingeloggt.

Feld	Inhalt
Use Case	Wohnungsanzeige bearbeiten
Akteur	Anbieter
Beschreibung	Der Anbieter ändert Daten einer bestehenden Anzeige.
Auslöser (Stimulus)	Anbieter wählt eine seiner Wohnungsanzeigen und klickt „Bearbeiten“.

Feld	Inhalt
Reaktion (Response)	1. System lädt die aktuellen Daten der Wohnungsanzeigen. 2. Anbieter ändert Informationen. 3. System validiert Änderungen. 4. System aktualisiert den Datensatz persistent. 5. System bestätigt die Änderung.
Comment	Vorbedingung: Anbieter ist eingeloggt und ist Ersteller der Wohnungsanzeige.

Feld	Inhalt
Use Case	Wohnungsanzeige löschen
Akteur	Anbieter
Beschreibung	Der Anbieter entfernt eine Wohnungsanzeige dauerhaft oder deaktiviert sie.
Auslöser (Stimulus)	Anbieter klickt bei einer Wohnungsanzeige auf „Löschen“.
Reaktion (Response)	1. System fordert Bestätigung an („Sind Sie sicher?“). 2. Anbieter bestätigt. 3. System löscht den Eintrag aus der Datenbank (oder setzt Status auf inaktiv). 4. System aktualisiert die Listenansicht.
Comment	Vorbedingung: Anbieter ist eingeloggt und ist Ersteller der Wohnungsanzeige.

3. Funktionale Anforderungen

3.1. Muss-Kriterien

- **M-01:** Das System muss fähig sein, zwischen den Benutzerrollen „User“ und „Anbieter“ zu unterscheiden.
    - Rolle: Anbieter
        - **M-02:** Das System muss dem Anbieter die Möglichkeit bieten, einen Account mit der Rolle „Anbieter“ zu registrieren.
        - **M-03:** Das System muss dem Anbieter die Möglichkeit bieten, eine neue Wohnungsanzeige zu erstellen.
        - **M-04:** Das System muss dem Anbieter die Möglichkeit bieten, seine eigenen Wohnungsanzeigen zu bearbeiten.
        - **M-05:** Das System muss dem Anbieter die Möglichkeit bieten, seine eigenen Wohnungsanzeigen zu löschen oder zu deaktivieren.
    - Rolle: User
        - **M-06:** Das System muss dem User die Möglichkeit bieten, einen Account mit der Rolle „User“ registrieren.
        - **M-07:** Das System muss dem registrierten User die Möglichkeit bieten, bestehende Wohnungsanzeigen zu bewerten.
        - **M-08:** Das System muss dem registrierten User die Möglichkeit bieten, die eigenen Bewertungen zu bearbeiten.
        - **M-09:** Das System muss dem registrierten User die Möglichkeit bieten, die eigenen Bewertungen zu löschen.
        - **M-10:** Das System muss fähig sein, Bewertungen ohne personenbezogene Daten (wie den Namen des Users) anzuzeigen.
- **M-11:** Das System muss dem unregistrierten User die Möglichkeit bieten, die Details von Wohnungsanzeigen anzusehen.
- **M-12:** Das System muss dem unregistrierten User die Möglichkeit bieten, abgegebene Bewertungen einzusehen.
- **M-13:** Das System muss einem registrierten User die Möglichkeit bieten, eine Bewertung mit einem Freitext-Kommentar zu ergänzen.
- **M-14:** Das System muss dem User die Möglichkeit bieten, die Nützlichkeit von fremden Bewertungen zu bewerten („War dies hilfreich?“).
- **M-15:** Das System muss dem User die Möglichkeit bieten, eine Wohnungsanzeige anhand der Adresse zu suchen (bzw. zu finden).
- **M-16:** Das System muss dem User die Möglichkeit bieten, eine Wohnungsanzeige über einen Link (z. B. E-Mail, Messenger) zu teilen.
- **M-17:** Das System muss registrierten Usern und Anbietern die Möglichkeit bieten, sich am System anzumelden (Login).

3.2. Soll-Kriterien

- **S-01:** Das System soll fähig sein, die Bewertungen thematisch (z. B. nach Lautstärke oder Helligkeit) zu gliedern.

- **S-02:** Das System soll dem User die Möglichkeit bieten, zwei Wohnungsanzeigen direkt miteinander zu vergleichen.
- **S-03:** Das System soll dem Anbieter die Möglichkeit bieten, die Anzahl der Aufrufe seiner Wohnungsanzeigen einzusehen.
- **S-04:** Das System soll dem registrierten User die Möglichkeit bieten, interessante Wohnungsanzeigen in einer persönlichen Merkliste zu speichern.
- **S-05:** Das System soll fähig sein, die gefundenen Wohnungsanzeigen auf einer interaktiven Karte darzustellen.

3.3. Kann-Kriterien

- **K-01:** Das System kann fähig sein, dem User für jede abgegebene Bewertung eine definierte Anzahl an Punkten gutzuschreiben.
- **K-02:** Das System kann dem User bei Erreichen einer definierten Anzahl an Bewertungen (z. B. 5, 10, 50) automatisch ein Meilenstein-Abzeichen verleihen.
- **K-03:** Das System kann fähig sein, dem User spezielle Themen-Badges (z. B. „Lärm-Experte“, „Helligkeits-Prüfer“) zuzuweisen, wenn er wiederholt entsprechende Kriterien bewertet hat.
- **K-04:** Das System kann fähig sein, dem User basierend auf der gesammelten Gesamtpunktzahl einen Level-Status (z. B. „Level 1 Entdecker“) zuzuweisen.

3.4. Abgrenzungskriterien

- Das System wird dem User keine Möglichkeit bieten, Mietverträge abzuschließen oder Zahlungen abzuwickeln.
- Das System wird den Usern keine Möglichkeit bieten, direkt miteinander zu kommunizieren (Chat).
- Das System wird nicht fähig sein, sich automatisch mit externen Immobilienportalen zu synchronisieren.

4. Nicht Funktionale Anforderungen

4.1. Anforderungen an die Datenhaltung

4.1.1. allgemeine Beschreibung der Daten

- **NFR-01 (Datenstruktur):** Es werden strukturierte Daten für Benutzerkonten (Stammdaten), Wohnungsanzeigen und Bewertungen relational gespeichert.
- **NFR-02 (Referenzielle Integrität):** Die Datenbank muss sicherstellen, dass Bewertungen immer zwingend mit einer existierenden Wohnung und einem existierenden Benutzer verknüpft sind (Foreign Key Constraints).
- **NFR-03 (Datenschutz):** Passwörter dürfen nicht im Klartext, sondern nur als Salted Hash (via Argon2 oder bcrypt) gespeichert werden.

4.1.2. Archivierung

- **NFR-04 (Löschkonzept):** Daten von gelöschten User-Accounts oder Anzeigen werden nicht archiviert, sondern physisch aus der Datenbank entfernt (DSGVO-Konformität: Recht auf Vergessenwerden), sofern keine gesetzlichen Aufbewahrungsfristen dagegen sprechen.

4.2. Anforderungen an die Benutzeroberfläche

4.2.1. allgemeine Anforderungen an die Oberfläche

- **NFR-05 (Technologie):** Die Benutzeroberfläche wird als Single Page Application (SPA) mit dem Svelte-Framework realisiert.
- **NFR-06 (Responsivität):** Das Design muss responsiv sein („Mobile First“-Ansatz) und sich dynamisch an Viewports von Smartphones (ab 375px Breite) bis zu Desktop-Monitoren anpassen.

4.2.2. Berechtigungen

Der Zugriff wird über rollenbasierte Berechtigungen (RBAC) gesteuert, die über **JWT (JSON Web Tokens)** validiert werden:

- **NFR-07 (Gast):** Nur Leserechte (Suche, Detailansicht).
- **NFR-08 (User):** Leserechte + Schreibrechte für eigene Bewertungen/Profil.
- **NFR-09 (Anbieter):** Leserechte + Schreibrechte für eigene Wohnungsanzeigen.
- **NFR-10 (Admin):** Volle Schreib- und Leserechte auf alle Daten zur Moderation.

4.2.3. individuelle Anpassung der Oberfläche

Entfällt für die erste Version (kein Dark-Mode oder Custom-Themes vorgesehen).

4.2.4. Bildschirmlayout

- **NFR-11 (Usability):** Das Layout orientiert sich an gängigen Standards (z. B. Material Design), um eine intuitive Bedienung zu gewährleisten. Wichtige Aktions-Buttons (Call-to-Action) müssen im sichtbaren Bereich (Above the Fold) platziert sein.

4.2.5. Prüfungen: feldbezogene und feldübergreifende

- **NFR-12 (Client-Side Validierung):** Eingabefelder im Svelte-Frontend müssen sich in Echtzeit validieren (z. B. E-Mail-Format, Pflichtfelder), bevor das Formular abgesendet wird.
- **NFR-13 (Server-Side Validierung):** Alle Eingaben müssen im Quarkus-Backend erneut validiert werden, um Manipulationen zu verhindern.

4.2.6. Drucklayout,Tastaturbelegung

Entfällt (Fokus liegt auf digitaler Nutzung).

4.2.7. Dialogstruktur, Dialogabläufe

- **NFR-14 (Sicherheitsabfrage):** Kritische Aktionen (z. B. „Anzeige löschen“) müssen durch einen Bestätigungsdialog („Sind Sie sicher?“) abgesichert werden.

4.2.8. Hilfesystem

Entfällt

4.3. Leistungsanforderungen

4.3.1. zeitbezogene oder umfangsbezogene Produktleistungen

4.3.2. Performance-Daten, Dialogantwortzeiten

- **NFR-15 (Ladezeit):** Das Laden der Startseite (First Contentful Paint) muss unter 2 Sekunden liegen (bei 4G-Verbindung).
- **NFR-16 (API-Latenz):** Suchanfragen müssen im Durchschnitt in unter 1 Sekunde vom Server beantwortet werden.
- **NFR-17 (Schreibgeschwindigkeit):** Das Speichern einer Bewertung darf nicht länger als 500 ms dauern.

4.3.3. Maximaler und durchschnittlicher Datenumfang bzw. Datendurchsatz

- **NFR-18 (Upload-Limit):** Bilder werden serverseitig auf maximal 5 MB pro Datei begrenzt.

- **NFR-19 (Text-Limit):** Bewertungen werden auf maximal 2000 Zeichen begrenzt, um Datenbank-Bloat und Spam zu verhindern.

4.3.4. Genauigkeit von Berechnungen

- **NFR-20 (Rundung):** Die Berechnung der Durchschnittsbewertungen (Sterne) erfolgt kaufmännisch gerundet auf eine Nachkommastelle (z. B. 4,3 Sterne)

4.4. Anforderung für Inbetriebnahme und Einsatz

4.4.1. Sicherheitsziele

- **NFR-21 (SQL-Injection):** Schutz durch konsequente Verwendung von Prepared Statements oder ORM (Hibernate/Panache) im Backend.

- **NFR-22 (XSS):** Schutz vor Cross Site Scripting durch automatisches Escaping im Svelte-Frontend.

- **NFR-23 (Verschlüsselung):** Die Kommunikation erfolgt ausschließlich verschlüsselt über HTTPS (TLS 1.2 oder höher).

4.4.2. Betriebssicherheit

Siehe Kapitel 4.4.1 und 4.4.5.

4.4.3. Installationsprozedur

- **NFR-24 (Containerisierung):** Das Deployment erfolgt automatisiert über Container-Images (Docker/Podman), definiert in einem `Compose`-File oder Kubernetes-Manifest.

4.4.4. Pilot- bzw. Probebetrieb

Entfällt

4.4.5. Fehlerreaktion, Garantie, Service, »Wiederanlauf«

- **NFR-25 (Fehlerseiten):** Bei Serverfehlern (HTTP 500) wird dem User eine generische Fehlerseite angezeigt; technische Details (Stacktrace) bleiben verborgen.
- **NFR-26 (Self-Healing):** Nach einem Absturz muss der Container-Orchestrator den Service automatisch neu starten.

4.4.6. Schulungen

Entfällt, da die Anwendung als selbsterklärende Web-Plattform für Endverbraucher konzipiert ist.

4.5. Qualitätsanforderungen

4.5.1. Qualitätsmerkmale

- **NFR-27 (Wartbarkeit):** Der Code muss modular aufgebaut sein (Trennung von API, Business-Logik und Datenbank-Layer).
- **NFR-28 (Portabilität):** Das System muss plattformunabhängig in jeder OCI-konformen Container-Laufzeitumgebung funktionieren.

4.5.2. Qualitätssicherung

- **NFR-29 (Statische Analyse):** Einsatz von Linter-Tools für Svelte und Java bei jedem Commit zur Sicherung der Code-Qualität.

4.5.3. Qualitätsnachweis

- **NFR-30 (Unit-Tests):** Automatisierte Tests müssen vor jedem Build in der CI-Pipeline erfolgreich durchlaufen werden.

4.5.4. Offenlegung der Qualitätskontrollpläne

- **NFR-31 (Protokollierung):** Testergebnisse der CI/CD-Pipeline (z. B. GitHub Actions Logs) dienen als automatischer Nachweis und werden protokolliert.

4.6. Anforderung an die Entwicklung

4.6.1. Entwurfsrestriktionen

4.6.2. Entwicklungs-Umgebung

- **Standard-IDE:** Visual Studio Code oder IntelliJ IDEA.
- **Laufzeitumgebung lokal:** Podman oder Docker Desktop..

4.6.3. Projekt-Organisation

- **Vorgehensmodell:** Agiles Vorgehen (angelehnt an Scrum/Kanban) mit wöchentlichen Sprints.

4.6.4. Projektplanung

- **Task-Management:** Verwaltung der Tasks und des Backlogs über ein Kanban-Board (GitHub Projects).

4.6.5. Projektüberwachung

Erfolgt durch regelmäßige Reviews im Rahmen der Lehrveranstaltung.

4.6.6. Projektsteuerung

Erfolgt durch regelmäßige Reviews im Rahmen der Lehrveranstaltung.

4.6.7. Konfigurationsmanagement

- **NFR-32 (Versionierung):** Quellcodeverwaltung zwingend über Git unter Verwendung von Feature-Branches.

4.6.8. Änderungsmanagement

- **NFR-33 (Datenbank-Migration):** Änderungen am Datenbankschema werden über Migrationstools (Flyway oder Liquibase) versioniert verwaltet.

4.6.9. Testanforderungen

- **Unit-Tests:** JUnit für Java-Komponenten.
- **Integrationstests:** Testen der API-Endpunkte gegen eine In-Memory-Datenbank (H2) oder Test-Container.

4.6.10. Reviews, Refactoring

- **NFR-34 (Vier-Augen-Prinzip):** Code-Changes werden nur per Pull Request nach einem erfolgreichen Review durch ein anderes Teammitglied gemerged.