A. Anforderungsspezifikation

1. Einleitung & Zielsetzung

Entwickelt werden soll eine nutzerfreundliche, intuitive App, die den Nutzenden das Verleihen ihrer eigenen und das Ausleihen fremder Bücher in ihrem lokalen Umfeld ermöglicht. Sie richtet sich insbesondere an Privatpersonen, die günstig und nachhaltig lesen und ggf darüber auch Gleichgesinnte treffen möchten. Die bisherigen Möglichkeiten zum Ausleihen von Büchern sind hauptsächlich städtische Büchereien, Unibibliotheken oder Bücherboxen. Büchereien und Unibibliotheken sind an Öffnungszeiten gebunden und in ländlicheren Gebieten nicht immer gut zu erreichen. Außerdem sind hier eine manchmal aufwendige Anmeldung und ein Jahresbeitrag erforderlich, was für einkommensschwache, bildungsfernere oder nicht deutschsprachige Haushalte ein weiteres Hindernis darstellen könnte. Bücherboxen (in denen Bücher abgegeben werden, die man selbst nicht mehr haben will) bieten meistens keine interessante Auswahl.

Die App bietet daher die Möglichkeit, unkompliziert und flexibel Bücher zu suchen, auszuleihen und selbst eigene anzubieten. Durch diesen Austausch können außerdem vor Ort Kontakte und ggf Communities entstehen, in denen man sich kennenlernen und weiter austauschen kann, z.B. bei Vorlieben für ein bestimmtes Genre oder eine bestimmte Sprache. Durch die klare zentrale Organisation in der App steigt auch die Bereitschaft für diesen Austausch, da der Verleih nicht an völlig fremde Menschen, sondern an andere registrierte Nutzer erfolgt und somit ggf auch rechtlich nachvollziehbar ist.

2. Rollen & Aktionen

- **Nutzende** als Ver- und Entleiher: Die Nutzenden sollen sich mit einem individuellen Nicknamen registrieren und identifizieren können. Sie können eigene Bücher, ggf. mit eigenen spezielleren Information wie Inhaltsangaben und Fotos einstellen, die Verfügbarkeit und die Konditionen der Ausleihe festlegen und den Ver- und Ausleihprozess komplett über die App abwickeln. Zudem sollen Bewertungen sowie der Austausch von Nachrichten für beide Seiten möglich sein.

- **Administratoren** für Verwaltung, Moderation, Troubleshooting, Konfliktvermittlung: Die Administratoren sollen die Nutzung der App beaufsichtigen, bei Konflikten schlichtend eingreifen und die Einhaltung der Nutzerregeln überwachen. Sie beantworten Fragen zur App und helfen bei technischen Problemen, können Inhalte bearbeiten oder löschen und ggf. auch Nutzende sperren.

3. Funktionale Anforderungen

3.1 Nutzerverwaltung
- Registrierung für neue/Login für registrierte Nutzer (Email & Passwort)
- Profilverwaltung: Neues Profil anlegen, Nickname wählen, persönliche Daten (Name, Email, Adresse) eingeben und ändern, Sichtbarkeitspräferenzen einstellen
- Passwort wählen, ändern, zurücksetzen

3.2 Bücherverwaltung
- Eigene Bücher zum Verleih einstellen: Titel, Autor, Genre, Inhaltsangaben, Fotos (Zustand)
- Angabe mind. eines Abholorts: Straße und Hausnr oder allgemeiner Ort (zB Café), PLZ & Stadt
- Jeweilige Verfügbarkeit angeben: Ab wann und bis wann kann das Buch ausgeliehen werden?
- Buch ändern (z.B. Zustand) oder löschen
- Überblick über eigene Transaktionen anzeigen lassen
- Buch als verfügbar/reserviert/verliehen markieren
- Buch nach Rückgabe wieder einstellen

3.3 Verleihen
- Leihanfragen sehen und beantworten/annehmen
- Verleihen/Rückgabe bestätigen
- Nach Rückgabe ggf. Zustand dokumentieren (Foto)

3.4 Büchersuche
- Bücher anhand von Titel, Autor, Genre oder Standort suchen
- Anzeigen, ob Buch verfügbar ist bzw ab wann
- Leihanfragen stellen
- Hinzufügen von (nicht verfügbaren) Büchern zur Wunschliste
- Anzeigen von Buch- und Nutzerinfos (z.B.: Wie lange verfügbar?)

3.5 Ausleihe & Rückgabe
- Ausleihe & Rückgabe bestätigen
- Zustand dokumentieren
- Rückgabedatum anzeigen lassen

3.6 Kommunikation
- Nachrichten senden & empfangen
- Benachrichtigungen erhalten (z. B. über neue Bewertung oder Anfrage)
- Konflikt melden

3.7 Bewertungen
- Erhaltene Bewertungen & Gesamtbewertung sehen
- Eigene Bewertungen verfassen
- Bewertungen anderer Nutzer lesen

4. Nicht-funktionale Anforderungen

- Benutzerfreundlichkeit: Übersichtliches Menü, klare Navigation, responsives Design
- Konformität mit DSGVO bei Nutzerdatenspeicherung
- Sicherheit von Authentifizierung und Nutzerdaten, Verschlüsselung sensibler Informationen
- Performance: Kurze Lade- und Antwortzeiten bei Aktionen
- Verfügbarkeit: Maßnahmen für Backup und Wiederherstellung, hohe Uptime
- Wartbarkeit: System sollte einfach an wachsende Nutzerzahlen angepasst werden können
- Plattformübergreifende Nutzbarkeit, sowohl per Web-App als auch auf Mobilgeräten

5. Zusammenfassung und Ausblick

Diese Anforderungsspezifikation legt den Entwicklungsgrundstein für eine nutzerfreundliche und funktionale Anwendung, die im lokalen Umfeld den unkomplizierten Austausch von Büchern ermöglicht, nachhaltiges Lesen fördert und sozialen Kontakt mit Gleichgesinnten erleichtert. Durch die klare Definition und Verteilung der Rollen, die umfassenden Verwaltungs-, Kommunikations- und Bewertungsfunktionen und die Berücksichtigung nicht-funktionaler Anforderungen kann eine stabile und skalierbare Plattform geschaffen werden.

In einem agilen Entwicklungsprozess sollten im nächsten Schritt die Anforderungen umgesetzt und dabei, sobald möglich, kontinuierlich Nutzerfeedback integriert werden. Besonders wichtig ist außerdem die rechtzeitige Planung und Durchführung von Testphasen und eine regelmäßige Überprüfung der Einhaltung von (datenschutz-)rechtlichen Vorgaben.

B. Problemstellung

Entwickelt werden soll die Datenbank für diese Anwendung. Die große Herausforderung für mich war vor allem die Abwägung zwischen einer möglichst flexiblen Datenbankstruktur auf der einen und einem möglichst geringen Redundanzrisiko auf der anderen Seite. Die Datenbank sollte zulassen, dass Daten präzise und an den jeweiligen Fall angepasst abgelegt und verwaltet werden können, dabei aber wartbar und sicher bleiben, vor allem vor dem Hintergrund möglicherweise ansteigender Nutzerzahlen. Dazu musste ich die Entscheidung treffen, ob ich in einzelnen Fällen eher mit weniger und dafür spezifischeren Fremdschlüsseln arbeite oder ob ich eher mehrere schmalere Fremdschlüssel nutze, die mehr Daten ablegen, aber auch flexiblere Aktionen ermöglichen.
Vor diesem Hintergrund war die Auswahl der geeigneten Entitäten und die korrekte Modellierung der Beziehungen für mich schwierig, insbesondere zwischen den Tabellen user, book, location und available und bei der conflict-Tabelle. Auch hier stellte sich die Frage nach der geeigneten Referenzierung und der Anzahl der joins, um die Struktur möglichst klar und frei von Inkonsistenzen zu halten und gleichzeitig flexibel zu bleiben.

C. Lösungsansätze Datenbank & Entity-Relationship-Modell

Um dieses Problem zu lösen, habe ich zunächst eine Auflistung der wichtigsten Fragen zur Datenmodellierung erstellt. Da die App in erster Linie eine unkomplizierte und nutzerfreundliche Möglichkeit zum Austausch zwischen Privatpersonen bieten soll, muss sie aus meiner Sicht in erster Linie zuverlässig und einfach bedienbar sein. Hierzu sollten Wartung und regelmäßige Updates einfach durchzuführen sein, um auf lange Sicht eine stabile Nutzererfahrung mit wenig Einbrüchen zu gewährleisten, die App sollte und flüssig laufen und die Gefahr von Abstürzen oder Fehlern sollte so gering wie möglich gehalten werden. Da es sich um eine kostenfrei nutzbare Anwendung handeln soll, sind bei guter Bewertung auch steigende Nutzerzahlen zu erwarten, was nicht zu Einbrüchen führen sollte, die Datenbank muss also leicht zu erweitern sein und Inkonsistenzen müssen so weit es geht vermieden werden.
Um diese Anforderungen zu erfüllen, habe ich mich für eine möglichst stabile, sichere und wartbare Struktur mit wenig Redundanzrisiko entschieden und die Entwicklung meiner Tabellen und ihrer Beziehungen an dieser Vorgabe ausgerichtet. Die Datenbank entspricht außerdem der 3. Normalform.

Die Tabelle user sammelt die erforderlichen Nutzerdaten und referenziert die address-Tabelle, da die Adressangabe lediglich der Sicherheit der App und der Nachvollziehbarkeit der Nutzerdaten in eventuellen Konfliktsituationen dienen soll; sie soll für andere Nutzer nicht sichtbar sein und insbesondere nicht zwingend zur Abwicklung von Leihvorgängen genutzt werden müssen. Daher hat jeder Nutzer nur einen Adressdatensatz, eine Adresse kann generell zu mehreren Nutzern gehören.

Bei der Verknüpfung der Tabellen user und book habe ich mich für eine n:m-Beziehung zwischen den beiden Tabellen entschieden, mit den allgemeinen Informationen zum Buch (wie Titel, Verlag und ISBN) in der book-Tabelle und dem konkret angebotenen Einzelexemplar in der user_book-Tabelle. Durch die Trennung werden doppelte Datensätze vermieden und die auf alle gleichen Bücher zutreffenden Informationen zentral zusammengefasst, so können sie einfacher bearbeitet, Inkonsistenzen vermieden und Speicherplatz gespart werden. Gleichzeitig werden in den späteren möglichen Vorgängen nur noch die konkreten Exemplare über user_book referenziert, indem die Tabellen location und user_book (Beziehung n:m) in der Zwischentabelle user_book_location verknüpft werden. Auf diese Weise soll es möglich sein, das entsprechende Buch auch an mehreren Orten zur Abholung anzubieten, so dass es auch dann in der Suche erscheint, wenn nur einer dieser Orte im gewünschten Radius liegt. Die location-Tabelle soll außerdem zulassen, dass optional entweder eine Straße mit Hausnummer oder ein sonstiger Ort wie zB ein Café angegeben werden kann. Auf diese Weise fällt die Angabe und Nachvollziehbarkeit des Ortes leichter, wenn man sich zB auf dem Weg zur Arbeit irgendwo treffen möchte, und erhöht so die Bedienbarkeit. Die Möglichkeit, das Buch zu versenden, wurde nicht ausdrücklich modelliert, da es sich um eine

unentgeltliche Leihe handeln soll – der Versand müsste zunächst von der verleihenden Person gezahlt werden und die andere Person müsste das Geld zurückzahlen, das erhöht die Gefahr für Konflikte zwischen den Nutzern. Bei eigener Bereitschaft kann in der Spalte user_book.notes eine Angabe dazu gemacht werden, zudem besteht die Möglichkeit für eine allgemeine Angabe dazu in der „about_me“-Spalte der user-Tabelle und für eine Absprache über die Nachrichten.
Die Tabelle available wurde zunächst als Fremdschlüssel in der user_book-Tabelle referenziert. Damit wäre aber eine eine abweichende Verfügbarkeit an verschiedenen Abholorten nicht mehr möglich gewesen, so dass ich die available_ID als Fremdschlüssel schließlich in der user_book_location-Tabelle referenziert habe.
Die request-Tabelle enthält den Fremdschlüssel user_book_loc_ID, da dieser Datensatz alle für die Anfrage nötigen Informationen enthält, und die loan-Tabelle enthält den Fremdschlüssel request_ID. So wird eine Mehrfachnutzung zu vieler Fremdschlüssel vermieden, um schnellere Abfragen zu ermöglichen und das Redundanzrisiko zu verringern.
Die Tabellen author und genre wurden jeweils als eigene Entitäten aufgebaut und zur book-Tabelle als n:m-Beziehungen modelliert, um die Normalform einzuhalten und gleichzeitig autoren- oder genrespezifische Suchanfragen und mehrfache Zuordnungen zuzulassen.

Die Verknüpfung der Bewertungstabelle mit einem konkreten Ausleihvorgang als optional ermöglicht es, dass sich eine Nutzerbewertung zwar auf einen „loan“ beziehen kann, aber auch reine Konversationen ggf bewertet werden können, falls zB ein Nutzer aufdringlich oder unfreundlich wird. Für diesen Fall sind die Attribute „reliability“ und „condition_as_described“ als optional eingetragen, „friendly_contact“ ist obligatorisch. Die Beschränkung der Bewertung auf den Kontakt zum anderen Nutzer und die Zustandsbeschreibung erfolgt mit der Absicht, dass in der Anwendung nicht die Bücher selbst bewertet sollen. Da der Ausleihvorgang unentgeltlich ist und man anders als bei einem Kauf kein finanzielles Risiko eingeht, soll so die Möglichkeit erhöht werden, dass man sich von den Meinungen anderer Leser nicht beeinflussen lässt. Außerdem kann man in der Spalte „notes“ bei Bedarf noch einen Kommentar abgeben.

Durch die conflict-Tabelle können Unstimmigkeiten zwischen den Nutzern gemeldet und ggf geklärt werden. Sie referenziert als Fremdschlüssel mit „opponent_ID“ die user_ID des Nutzers, mit dem ein Konflikt aufgetreten ist, und mit „reported_by_ID“ die user_ID des meldenden Nutzers. Die optionale Referenz auf die loan-Tabelle soll den Konflikt an einen konkreten Ausleihvorgang knüpfen können, aber auch anderweitige Meldungen zulassen.
Die Tabelle photo referenziert als optionale Fremdschlüssel die Tabellen user_book, loan und conflict, damit der verleihende Nutzer bei Bedarf Bilder zentral speichern und im Zweifel für mehrere Ausleihvorgänge verwenden kann, so es keine doppelte Speicherung gibt. Gleichzeitig soll ein Nutzer beim Ausleihen eines fremden Buchs bei Bedarf den Zustand des geliehenen Buchs dokumentieren können, und das ist nur bei der loan-Tabelle sinnvoll. Und zuletzt soll bei Bedarf die Konflikt-Tabelle referenziert werden können.

Vor dem Hintergrund, dass Datenintegrität und Stabilität priorisiert werden sollen, habe ich mich beim Umgang mit gelöschten Elementen für die Verwendung von ON DELETE CASCADE und ON DELETE SET NULL entschieden. Ersteres verwende ich dabei, wenn alle von dem gelöschten Element abhängigen Daten ebenfalls entfernt werden können (weil die referentielle Integrität nicht gefährdet wird) und sollen (weil sie ohne das gelöschte Element keinen Nutzen mehr haben). ON DELETE SET NULL wird genutzt, wenn die referentielle Integrität bei Löschung der abhängigen Datensätze gefährdet wäre und diese noch gebraucht werden. So sollen beispielsweise Bewertungen anderer Nutzers nicht automatisch mit verschwinden, wenn ein Nutzer sein Konto löscht. Durch die Unterscheidung zwischen beiden Anwendungsfällen wird gewährleistet, dass die Datenbank keinen „Datenmüll“ enthält, während erforderliche Daten bewahrt werden.

Zur Umsetzung der Entwicklung habe ich mich für MySQL entschieden, weil es stabil und weit verbreitet ist und sich wegen seiner Zuverlässigkeit gut für die Arbeit mit (wenn auch fiktiven) Nutzerdaten eignet. Außerdem funktionieren die Abfragen schnell, was ich für einen Vorteil beim Testen halte, und man kann sich auch als Anfänger gut einarbeiten, um sich auf die Problemlösung beim Entwickeln der Datenbank zu konzentrieren.