Contents

- Section 1 - Project Description
  - 1.1 Project
  - 1.2 Description
  - 1.3 Revision History
- Section 2 - Overview
  - 2.1 Purpose
  - 2.2 Scope
  - 2.3 Requirements
- Section 3 - System Architecture
- Section 4 - Data Dictionary
- Section 5 - Software Domain Design
  - 5.1 Software Application Domain Chart
- Section 6 – Data Design
- Section 7 - User Interface Design
- Section 8 – References
- Section 9 – Glossary

Section 1 - Project Description

1.1 Project

Project Name: Neighborhood Library Service
Project Type: Web-Based Library Management System
Technology Stack: Python Backend (REST), PostgreSQL, React

1.2 Description

The Neighborhood Library Service is a digital system designed to manage books, members, and lending operations for a small-to-medium neighborhood library.

The system replaces manual tracking with a secure, scalable, and auditable platform that:

- Tracks book inventory
- Manages library members
- Records borrowing and return transactions
- Enables query and reporting of lending data
- Ensures transactional integrity and operational efficiency

The architecture follows enterprise design principles, including layered architecture, transactional consistency, and future scalability readiness.

1.3 Revision History

Date	Comment	Author
2026-02-23	Initial Architecture Document	Chaitenya Yadav

Section 2 - Overview

2.1 Purpose

The purpose of this document is to define:

- System functionality
- Technical architecture
- Data model
- Design decisions
- Interface contracts
- Operational considerations

This document serves as a reference for development, QA, and stakeholders.

2.2 Scope

In Scope

- Book management
- Author Management
- Publisher Management
- Member management
- Borrow & return tracking
- Overdue tracking
- Basic reporting

Out of Scope

- Payment gateway integration
- Multi-branch distributed inventory
- External ERP integration

2.3 Requirements

Functional Requirements

ID	Requirements

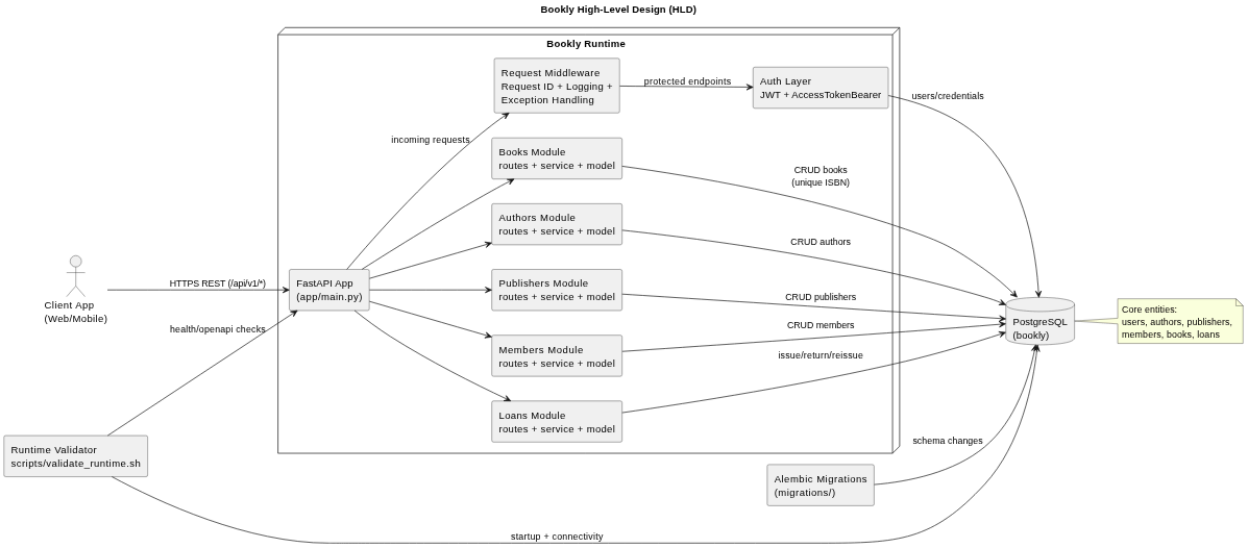
FR-01	Books CRUD Operations
FR-02	Authors CRUD Operations
FR-03	Publisher CRUD Operations
FR-04	Members CRUD Operations
FR-05	Record book loan (borrow) transaction
FR-06	Record return transaction
FR-07	Record reissue transaction
FR-08	Query borrowed books
FR-09	Prevent double borrowing
FR-10	User CRUD Operations

Non-Functional Requirements

ID	Requirements
NFR-01	ACID-compliant transactions
NFR-02	API response (latency) < 200ms
NFR-03	Secure input validation
NFR-04	Scalable stateless backend
NFR-05	Database integrity via constraints

Section 3 - System Architecture

Refer plantUml file “design/bookly_hld.puml” for more detail



Key Architectural Decisions:

- Stateless backend
- Separation of borrow records for audit
- Prepared for containerized deployment

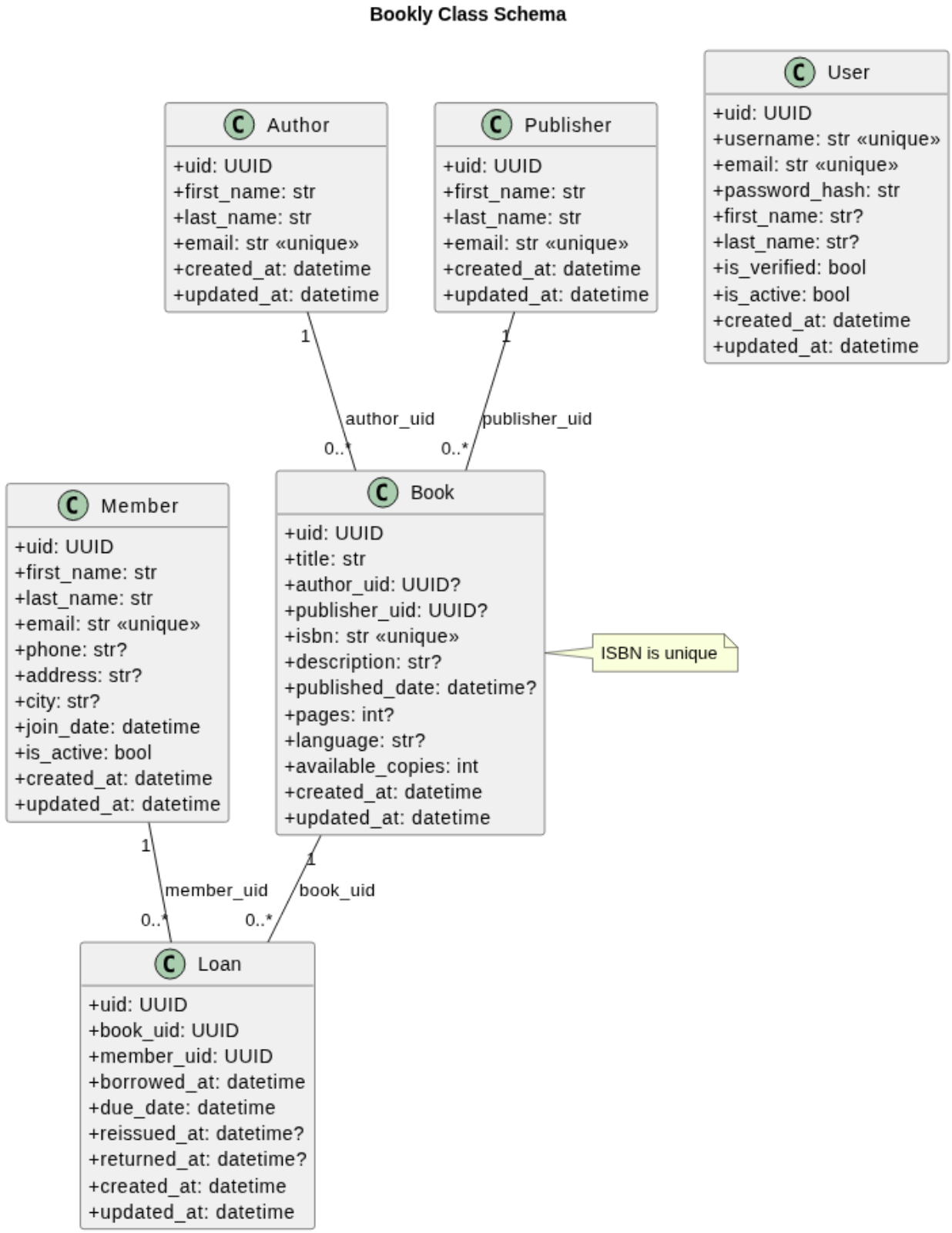
Section 4 - Data Dictionary

Refer to the ER Diagram for more details.

Section 5 - Software Domain Design

5.1 Software Application Domain Chart

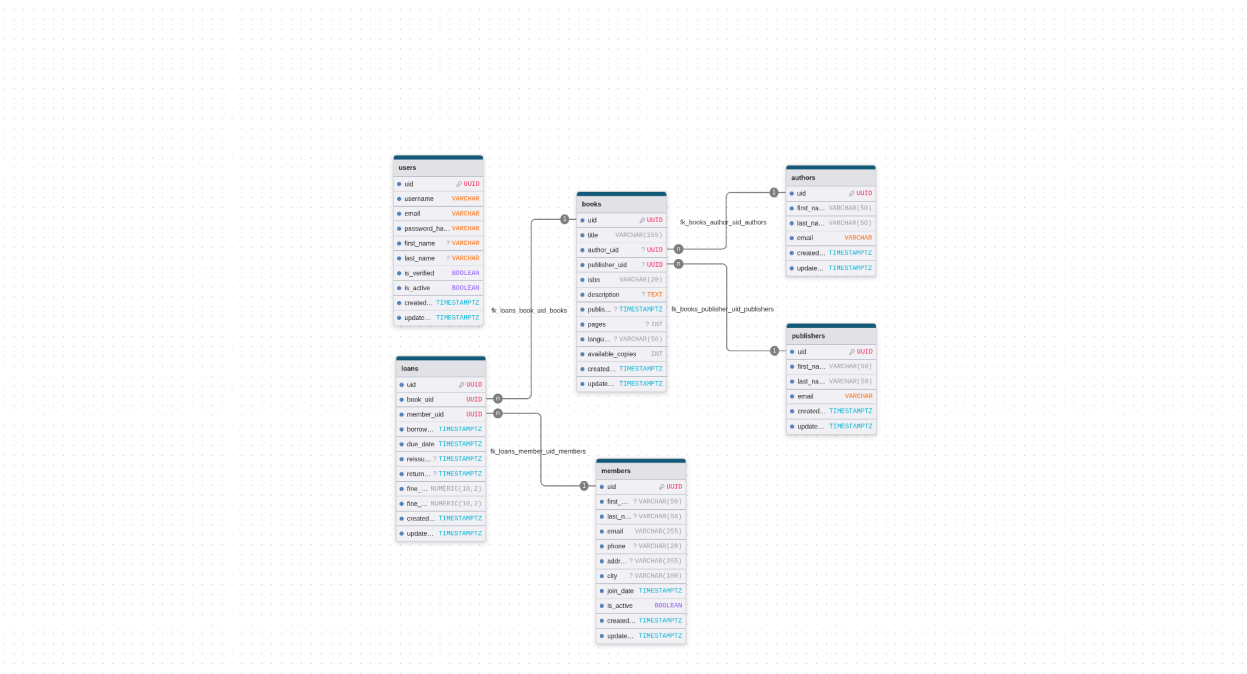
Refer plantUML file “design/bookly_schema.puml”


Bookly Class Schema
User
+uid: UUID
+username: str «unique»
+email: str «unique»
+password_hash: str
+first_name: str?
+last_name: str?
+is_verified: bool
+is_active: bool
+created_at: datetime
+updated_at: datetime
Author
+uid: UUID
+first_name: str
+last_name: str
+email: str «unique»
+created_at: datetime
+updated_at: datetime
Publisher
+uid: UUID
+first_name: str
+last_name: str
+email: str «unique»
+created_at: datetime
+updated_at: datetime
1
1
author_uid
publisher_uid
0..*
0..*
Member
+uid: UUID
+first_name: str
+last_name: str
+email: str «unique»
+phone: str?
+address: str?
+city: str?
+join_date: datetime
+is_active: bool
+created_at: datetime
+updated_at: datetime
Book
+uid: UUID
+title: str
+author_uid: UUID?
+publisher_uid: UUID?
+isbn: str «unique»
+description: str?
+published_date: datetime?
+pages: int?
+language: str?
+available_copies: int
+created_at: datetime
+updated_at: datetime
ISBN is unique
1
1
member_uid
book_uid
0..*
0..*
Loan
+uid: UUID
+book_uid: UUID
+member_uid: UUID
+borrowed_at: datetime
+due_date: datetime
+reissued_at: datetime?
+returned_at: datetime?
+created_at: datetime
+updated_at: datetime

Section 6 – Data Design

Refer DBML file “design/schema.dbml”

ER Diagram



Section 7 - User Interface Design

Refer API Doc link

Section 8 – References

Reference

Section 9 – Glossary

A

ACID
Atomicity, Consistency, Isolation, Durability — database properties that guarantee reliable transactions.

API (Application Programming Interface)
A contract that defines how software components communicate with each other.

ADR (Architecture Decision Record)
A document that captures important architectural decisions along with context and consequences.

B

Backend
Server-side application responsible for business logic and database interaction.

Borrow Record
A transactional record representing a book borrowed by a member.

C

CRUD
Create, Read, Update, Delete — basic data operations.

Concurrency Control
Mechanisms to ensure correct behavior when multiple transactions occur simultaneously.

D

E

ER Diagram (Entity Relationship Diagram)
A visual representation of database entities and their relationships.

F

FK (Foreign Key)
A database constraint that links one table to another.

Functional Requirement (FR)

A requirement that defines system behavior.

G

GUI (Graphical User Interface)
The visual interface users interact with.

H

HLD (High-Level Design)
Architecture-level system design document.

HTTP (HyperText Transfer Protocol)
Protocol used for communication between web clients and servers.

I

Index (Database Index)
A structure that improves database query performance.

Isolation Level
Defines visibility of transactions in concurrent operations.

J

JWT (JSON Web Token)
A secure token format used for authentication and authorization.

L

Layered Architecture
Architectural pattern dividing system into layers (UI, Service, Repository, DB).

M

Microservice
An independently deployable service responsible for specific functionality.

N

NFR (Non-Functional Requirement)
A requirement defining system qualities (performance, scalability, security).

O

ORM (Object-Relational Mapping)
A technique to map database tables to programming objects.

P

PK (Primary Key)
A unique identifier for each record in a database table.

PostgreSQL
Open-source relational database used as the system data store.

Protocol Buffers (Protobuf)
Serialization format used by gRPC.

R

REST (Representational State Transfer)
Architectural style for building web services.

Repository Layer
Layer responsible for database interaction.

RBAC (Role-Based Access Control)

Authorization model based on user roles.

S

SLA (Service Level Agreement)
Defines service availability and performance guarantees.

SLO (Service Level Objective)
Target performance metric within an SLA.

Stateless
No client session data stored on server between requests.

SQL Injection
Security vulnerability allowing malicious SQL execution.

T

Transaction
A unit of work executed atomically in a database.

Traceability Matrix
Mapping between requirements and implementation/testing artifacts.

U

UUID (Universally Unique Identifier)
A unique identifier used for primary keys.

V

Validation
Process of ensuring input data meets defined rules.

W

Web Client
Frontend application interacting with backend services.