BookBuddy

1. System Interaction and Architecture Description

BookBuddy is a Spring Boot–based web application designed to operate within a standard client–server environment. The system's backend runs on the Java Virtual Machine (JVM) using Java 21 and the Spring Boot 3 framework. It requires the Maven build tool for dependency management and the Jakarta Persistence API (JPA) for database integration.

The application interacts with a MySQL database, which is automatically configured and generated by Hibernate (the default JPA implementation). No manual SQL setup is required, as entity mappings are used to define table structures
BookBuddy is designed to run on any modern operating system (Windows, macOS, or Linux) capable of hosting a Java runtime environment and MySQL server.

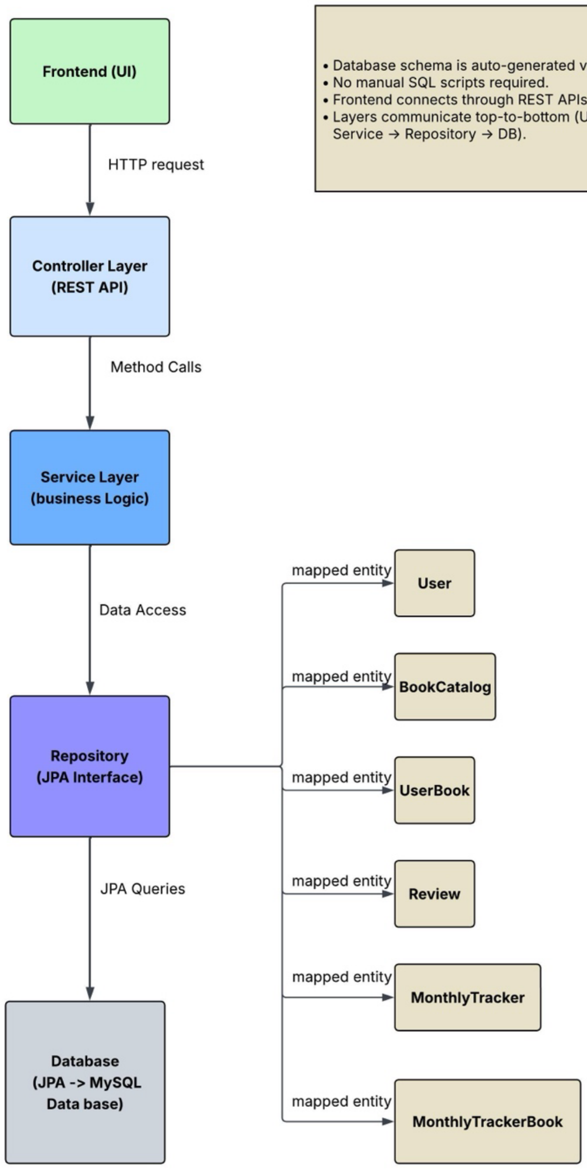
2. System Architecture

BookBuddy follows a layered Spring Boot MVC architecture, which divides the system into independent components responsible for different concerns. Each layer communicates downward, maintaining a clear separation of responsibilities and minimizing coupling between components.

At the top, the UI Layer handles user interaction through web pages and sends HTTP requests to the Controller Layer, which defines REST endpoints for user, book, review, and tracking operations. Controllers delegate application logic to the Service Layer, which validates data and applies business rules.

The Repository Layer manages data access using JPA repositories, which are automatically implemented by Spring. These repositories interact with Hibernate, which maps the entity classes to relational database tables stored in a MySQL database.

This modular design supports scalability and maintainability, allowing each layer to evolve independently without affecting others.



- Database schema is auto-generated via JPA/Hibernate.
- No manual SQL scripts required.
- Frontend connects through REST APIs (Spring Controllers).
- Layers communicate top-to-bottom (UI → Controller → Service → Repository → DB).

Controller Layer

UserController
BookCatalogController
UserBookController
Review Controller
MonthlyTrackerBookController
MonthlyTrackerController

Service Layer

User Service
BookCatalogService
UserBookService
ReviewService
MonthlyTrackerService
MotnthlyTrackerBookServie

Repository

UserRepository
BookCatalogRepository
UserBookRepository
ReviewRepository
MonthlyTrackerRepository
MonthlyTrackerBookRepository

Model Layer(Entities)

User
BookCatalog
UserBook
Review
MonthlyTracker
MonthlyTrackerBook

3. System Decomposition

The architecture is decomposed into five primary packages, each corresponding to a major functional layer of the system:

Package / Layer	Classes	Description / Role in System
com.bookbuddy.model	User, BookCatalog, UserBook, Review, MonthlyTracker, MonthlyTrackerBook	Defines the core data model of the system. Each class represents an entity that is automatically mapped to a database table through JPA annotations. These classes encapsulate the system's business data and relationships between users, books, and reading goals.
(supporting enums)	Genre, Months, ShelfStatus	Supporting enumerations that define fixed values used by entities. Genre represents book categories, Months defines time periods for trackers, and ShelfStatus indicates the state of a user's book (reading, completed, or wishlist).
com.bookbuddy.repository	UserRepository, BookCatalogRepository, UserBookRepository, ReviewRepository, MonthlyTrackerRepository, MonthlyTrackerBookRepository	Provides data-access operations and persistence management using JPA. Each repository corresponds to an entity and extends JpaRepository, allowing standard CRUD operations without manual SQL queries.
com.bookbuddy.service	UserService, BookCatalogService, UserBookService, ReviewService, MonthlyTrackerService, MonthlyTrackerBookService	Implements business logic and validation. Each service coordinates data flow between controllers and repositories, applying application rules and ensuring consistent entity management.
com.bookbuddy.controller	UserController, BookCatalogController, UserBookController, ReviewController, MonthlyTrackerController, MonthlyTrackerBookController	Defines RESTful API endpoints for CRUD operations and system interaction. The controllers handle HTTP requests, invoke service methods, and return structured JSON responses to the client.
com.bookbuddy.config	SecurityConfig	Provides configuration for application security, CORS policy, and API documentation using Swagger. Supports integration and future deployment readiness.

4. Error and Exception Handling Strategy

BookBuddy includes multiple levels of error handling to ensure reliability and graceful recovery from anticipated issues.

- **Invalid input:** Managed through Jakarta Bean Validation annotations such as @NotBlank, @Email, and @Pattern. Invalid user data returns an HTTP 400 Bad Request response.
- **Entity not found:** Repository calls return Optional objects; if empty, the service layer throws custom exceptions handled by the controller.
- **Database or network failures:** Spring Boot automatically propagates exceptions as 500 Internal Server Error responses. Future versions (Sprint 2) will include a @ControllerAdvice class for centralized exception handling.
- **Frontend communication errors:** Handled through user-friendly alerts in the UI layer (implemented in Sprint 2).

These mechanisms collectively ensure the system responds predictably to both expected and unexpected failures.

CRC Cards

The following Class–Responsibility–Collaborator (CRC) cards describe the key model entities in the BookBuddy system. Each class defines a single responsibility and maintains clear relationships with its collaborators to promote high cohesion and low coupling.

Class Name: 'MonthlyTracker'	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: • Represent a user’s monthly reading goal (year, month, and target count). • Track progress and goal completion. • Manage related 'MonthlyTrackerBook' entries for detailed tracking. • Summarize completed and pending goals.	**Collaborators:** • 'User' – owner of the tracker. • 'MonthlyTrackerBook' – links books to the tracker. • 'MonthlyTrackerRepository', 'MonthlyTrackerService', 'MonthlyTrackerController'. • 'Months' – defines the active month.

Class Name: 'MonthlyTrackerBook'

Parent Class (if any): None
Subclasses (if any): None

Responsibilities:	**Collaborators:**
• Connect a 'MonthlyTracker' to individual 'UserBook' entries. • Record progress toward monthly reading goals (books read versus total). • Update completion state when goals are achieved. • Provide summaries for reporting or monthly resets.	• 'MonthlyTracker' – parent goal record. • 'UserBook' – books being tracked. • 'MonthlyTrackerBookRepository', 'MonthlyTrackerBookService', 'MonthlyTrackerBookController'.

Class Name: 'Review'

Parent Class (if any): None
Subclasses (if any): None

Responsibilities:	**Collaborators:**
• Store review text and star rating for a book. • Link each review to a specific user and book. • Support retrieval and sorting by user or book. • Contribute to average rating calculations within the service layer.	• 'User' – author of the review. • 'BookCatalog' – book being reviewed. • 'ReviewRepository', 'ReviewService', 'ReviewController'.

Class Name: 'UserBook'	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: • Represent the relationship between a 'User' and a 'BookCatalog' entry. • Store 'ShelfStatus' such as Reading, Completed, or Wishlist. • Record notes or progress for each book. • Allow updates to reading status or removal from a user's library.	**Collaborators:** • 'User' – owner of the record. • 'BookCatalog' – referenced book information. • 'UserBookRepository', 'UserBookService', 'UserBookController'. • 'ShelfStatus' – defines reading state.

Class Name: 'BookCatalog'	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: • Represent all books available in the catalog. • Store book metadata such as title, author, genre, and ISBN. • Support search and filtering by title, genre, or author. • Maintain connections to user-specific entries and reviews.	**Collaborators:** • 'UserBook' – links users to catalog entries. • 'Review' – stores user feedback and ratings. • 'BookCatalogRepository', 'BookCatalogService', 'BookCatalogController'. • 'Genre' – defines book categories.

Class Name: 'User'	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: • Store and manage user information such as first name, last name, username, email, and password. • Maintain the user's personal library through associated 'UserBook' records. • Provide methods to update or validate profile details. • Add or remove 'UserBook' entries. • Generate statistics such as total books owned and reading progress.	**Collaborators:** • 'UserBook' – manages the books owned by the user. • 'BookCatalog' – accessed indirectly through 'UserBook'. • 'MonthlyTracker' – tracks the user's reading goals. • 'UserRepository', 'UserService', 'UserController'.