Bookmark Web Application

: Assignment 2

Student: Morandau Felix-Aleaxndru

**Group: 30435**

Table of Contents

Contents

[1. Requirements Analysis 3](#_Toc192108411)

[1.1 Assignment Specification 3](#_Toc192108412)

[1.2 Functional Requirements 3](#_Toc192108413)

[1.3 Non-functional Requirements 3](#_Toc192108414)

[2. Use-Case Model 4](#_Toc192108415)

[3. System Architectural Design 4](#_Toc192108416)

# 1. Requirements Analysis

## Assignment Specification

The goal of this project is to create a bookmark management app. Users can register, log in, manage their personal bookmarks, and organize them into collections. Users can also browse/post collections (which can be structured as tutorials, guides and all that) and can search for them based on several filters. The app uses a Spring Boot backend and a React/TypeScript frontend with clean code, proper error handling, and input validation.

## Functional Requirements

User Management:

* Register new users with a unique username, valid email, and strong password.
* Update user profile details (password, email, and bio).
* Delete user accounts.

Bookmark Handling:

* Create and manage bookmark collections.
* Add and remove bookmarks from collections.

Data Retrieval:

* Get user details by ID, username, or email.
* Filter user/bookmarks/collections lists based on criteria.

Error Handling & Validation:

* Validate inputs on both the frontend and backend (e.g., check password strength and email format).
* Display clear error messages.

## Non-functional Requirements

Validation:

* Make sure all user inputs are checked for errors and follow the correct format.

Architecture:

* Use a layered design (controllers, services, repositories).

Database:

* Use an ORM for handling database operations for a cleaner and better structured code.

API Design:

* Build a RESTful API with proper HTTP methods and clear responses, including detailed error messages.

User Experience:

* Ensure the frontend is responsive and easy to use (not really the case).

# Use-Case Model

A diagram of a person

AI-generated content may be incorrect.

**Client Login / Register**

Actor: Client  
Preconditions: Client has not yet authenticated.  
Postconditions: Client is authenticated and has an active session.

Main Success Scenario:

1. Client navigates to the “Login/Register” screen.
2. Client chooses Register (new user) or Login (existing user).
3. Register: Client supplies username, email, password → System validates, creates new user record, and logs them in.
4. Login: Client supplies username & password → System validates credentials and starts a session.
5. System redirects client to their dashboard.

Extensions:

* 3a. Username or email already in use → System displays “User already exists” error.
* 4a. Invalid credentials → System displays “Incorrect username or password.”

**Create Bookmark**

Actor: Client  
Preconditions: Client is logged in.  
Postconditions: A new Bookmark exists and is associated with the Client.

Main Success Scenario:

1. Client clicks “New Bookmark.”
2. System displays a form for URL, Title, Description, Category.
3. Client fills in form and submits.
4. System validates that URL is well‑formed and Title is present.
5. System persists the Bookmark and shows a confirmation message.

Extensions:

* 4a. URL invalid → System highlights the field and displays “URL must be valid.”
* 4b. Title missing → System displays “Title is required.”

**Create Collection**

Actor: Client  
Preconditions: Client is logged in.  
Postconditions: A new Collection exists, owned by the Client.

Main Success Scenario:

1. Client selects “Create Collection.”
2. System presents a form for Name, Category, Description, Visibility (public/private).
3. Client completes and submits the form.
4. System validates mandatory fields (e.g. Name) and saves the Collection.
5. System confirms creation and lists the empty Collection.

Extensions:

* 3a. Name empty or duplicate → System displays “Name is required/duplicate.”

**Add Bookmarks to Your Collections**

Actor: Client  
Preconditions: Client is logged in; client has at least one Bookmark and at least one Collection.  
Postconditions: Selected Bookmark(s) are associated with the chosen Collection.

Main Success Scenario:

1. Client views a list of their Bookmarks.
2. Client clicks “Add to Collection” on one or more Bookmarks.
3. System shows a dropdown of the Client’s Collections.
4. Client selects a Collection and confirms.
5. System saves the association and shows a success message.

**Save Other Users’ Collections**

Actor: Client  
Preconditions: Client is logged in; there are public Collections created by other users.  
Postconditions: The chosen public Collection is added to the Client’s “Saved Collections.”

Main Success Scenario:

1. Client browses public Collections.
2. Client clicks “Save Collection” on a public Collection.
3. System adds that Collection to the Client’s saved list.
4. System confirms “Collection saved.”

Extensions:

* 2a. Collection is private → System disallows and shows “Cannot save private collection.”
* 3a. Already saved → System shows “You’ve already saved this.”

**View User / Bookmark / Collection**

Actor: Admin  
Preconditions: Admin is logged in.  
Postconditions: Admin is viewing the requested record(s).

Main Success Scenario:

1. Admin navigates to the management dashboard.
2. Admin selects “View Users,” “View Bookmarks,” or “View Collections.”
3. System displays a paginated list.
4. Admin can filter/search, then click an item to see details.

**Add User / Bookmark / Collection**

Actor: Admin  
Preconditions: Admin is logged in.  
Postconditions: A new User, Bookmark, or Collection record is created.

Main Success Scenario:

1. Admin clicks “Add User” (or Bookmark/Collection).
2. System presents the corresponding creation form.
3. Admin fills in all required fields and submits.
4. System validates and saves the new entity.
5. System confirms “Created successfully.”

**Update User / Bookmark / Collection**

Actor: Admin  
Preconditions: Admin is logged in; the entity exists.  
Postconditions: The selected record is updated in the system.

Main Success Scenario:

1. Admin locates the record via “View…” table.
2. Admin clicks “Edit.”
3. System shows a form populated with current data.
4. Admin modifies fields and submits.
5. System validates changes and persists them.
6. System confirms “Update successful.”

Export Collection

Actor: User

Preconditions: User is logged in and has a collection

Postconditions: A modal appears on the screen with the preferred format

Main Success Scenario

# System Architectural Design

## Architectural Pattern Description

The application follows a layered architecture with a clear separation of tasks for the backend. At the top, controllers handle incoming HTTP requests and forward them to the service layer, which contains the business logic. The service layer interacts with the repository layer where data access is abstracted using Spring Data JPA (Jakarta Persistence API) and Hibernate. The model layer defines the entities, while Hibernate serves as the Object-Relational Mapping (ORM) tool that maps these entities to the relational database via reflection. This design promotes maintainability, scalability, and easy testing by keeping the web, business, and persistence concerns separate.

## Diagrams

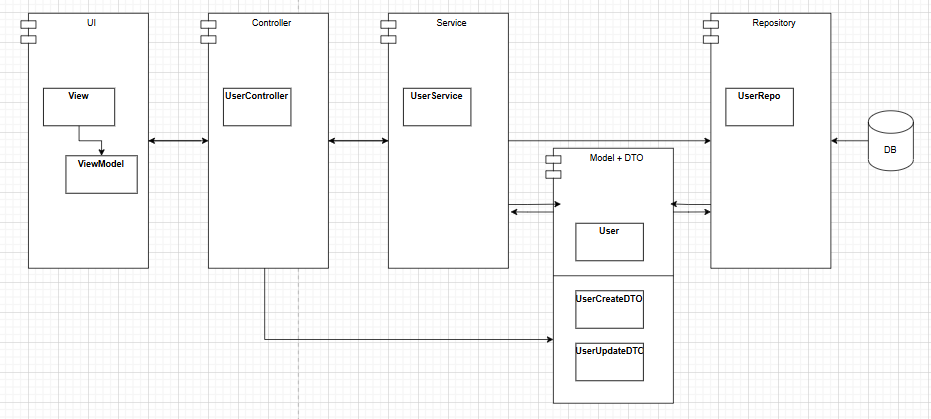
**A diagram of a software development

AI-generated content may be incorrect.**

This diagram shows the layered architecture for the application. The Frontend/UI is what users interact with, sending requests to the Controller layer, which provides RESTful endpoints. The Service layer contains the main business logic and decides how to handle each request. The Repository layer uses Hibernate (an ORM) and JPA to access and manage data in the Relational Database (PostgreSQL). By splitting these parts into layers, the code is easier to understand, maintain, and extend.

# Class Design

## Package + Class Diagram



# Data Model

A screenshot of a computer

AI-generated content may be incorrect.

This diagram shows a relational database for the bookmark application. The users table holds user details (username, email, password, etc.). A bookmark references a link (storing URL info) and a user (the owner) (practically a many to many between user and link so links should not be stored multiple times). A collection groups bookmarks together, and bookmark\_collection links bookmarks to specific collections (more bookmarks can be in more collections => many-to-many). The saved\_collections table tracks which users have saved which collections (many-to-many).