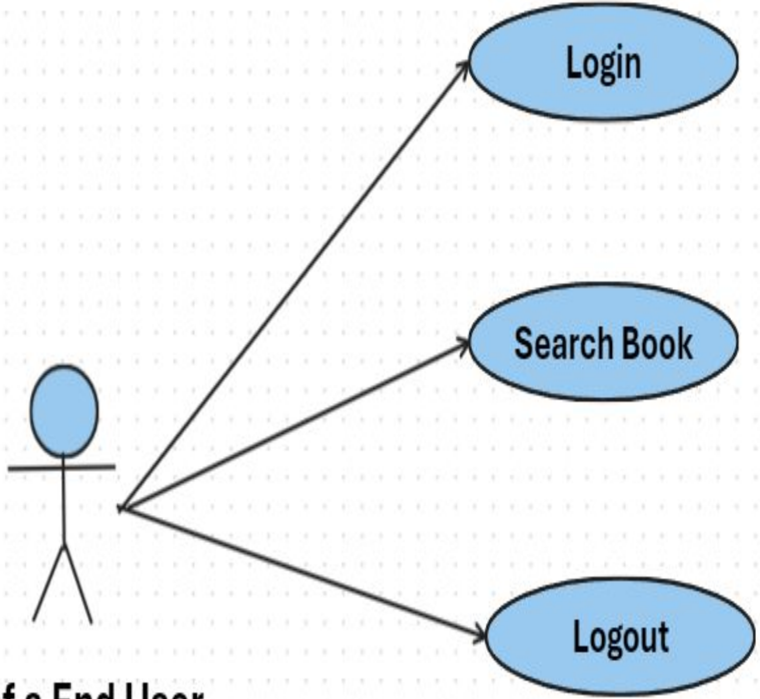
Library Management

A Book Search and Management System

Akhil Goud Burra

Project Description

- The Library Search and Management System is a web application designed to streamline the process of searching for books within a library's catalog and managing library operations.

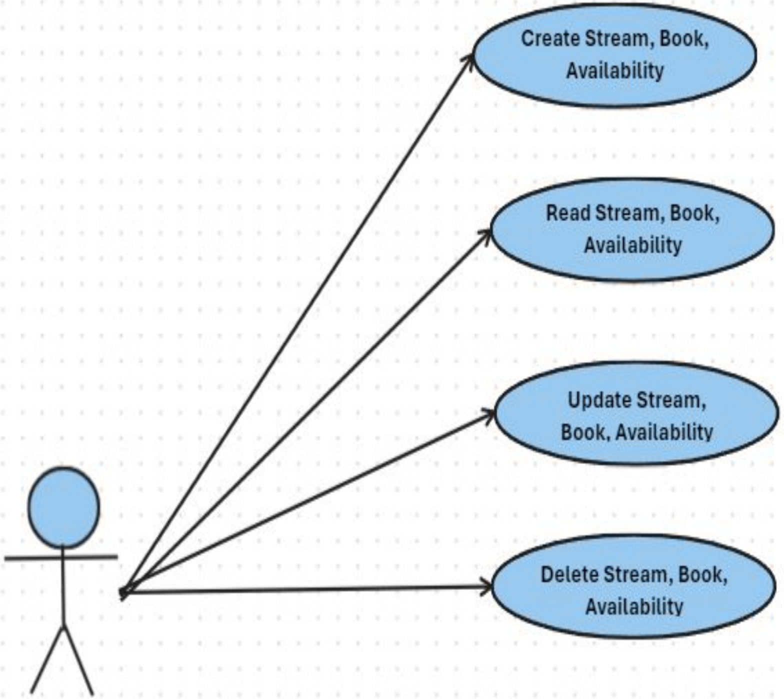


Login

Search Book

Logout

Role of a End User



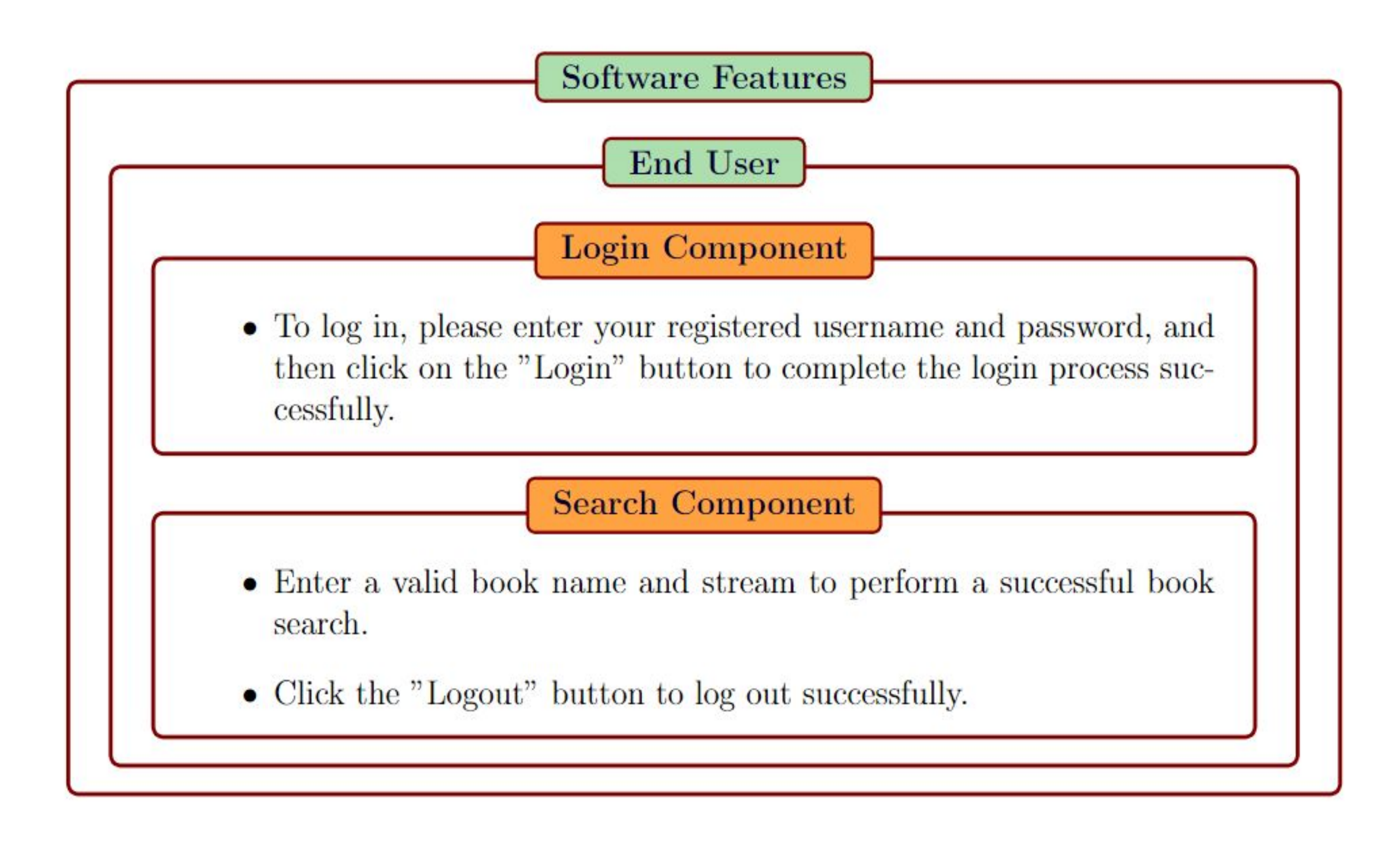
Create Stream, Book, Availability

Read Stream, Book, Availability

Update Stream, Book, Availability

Delete Stream, Book, Availability

Role of a Admin User



Software Features

End User

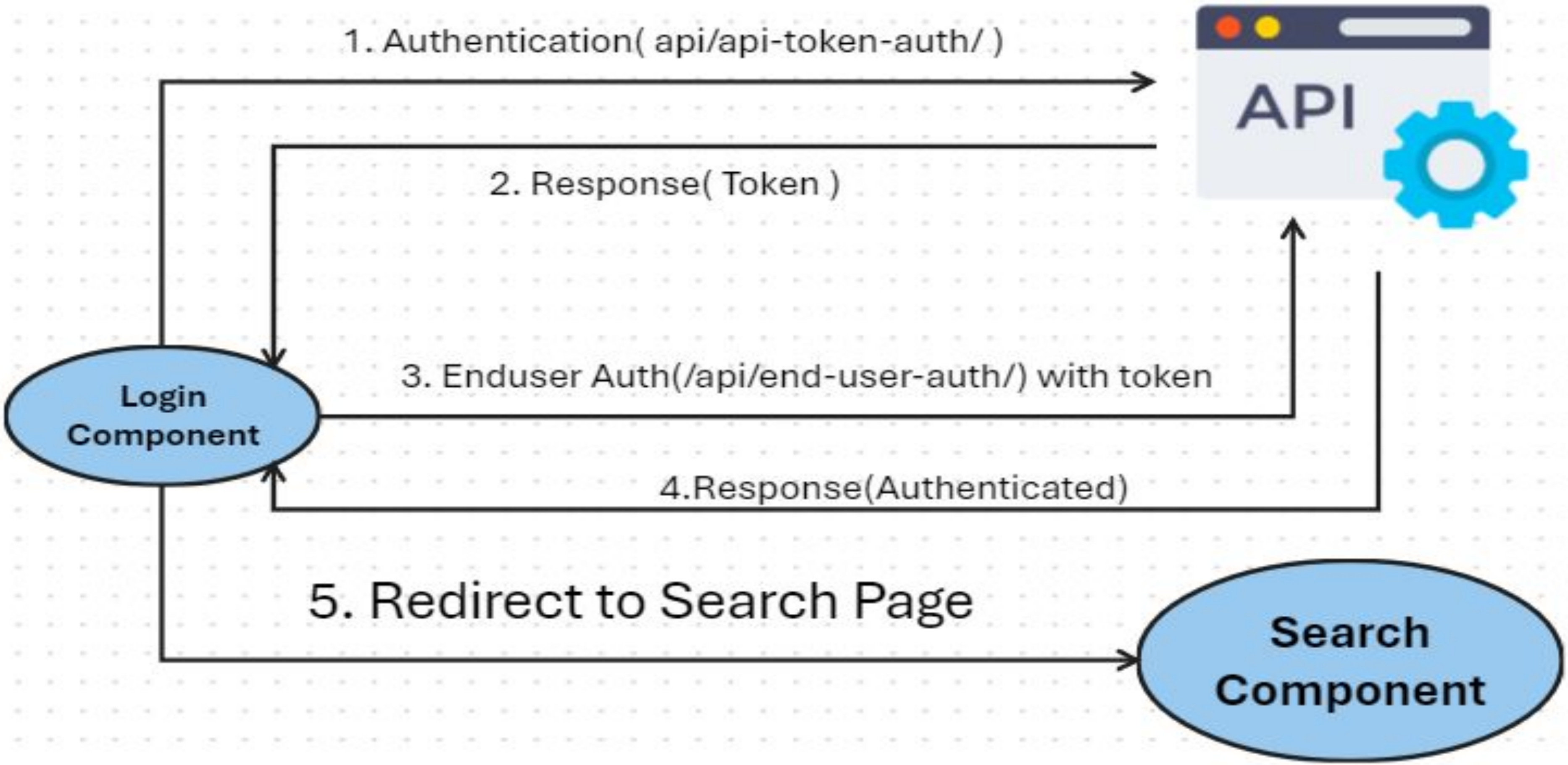
Login Component

- To log in, please enter your registered username and password, and then click on the "Login" button to complete the login process successfully.

Search Component

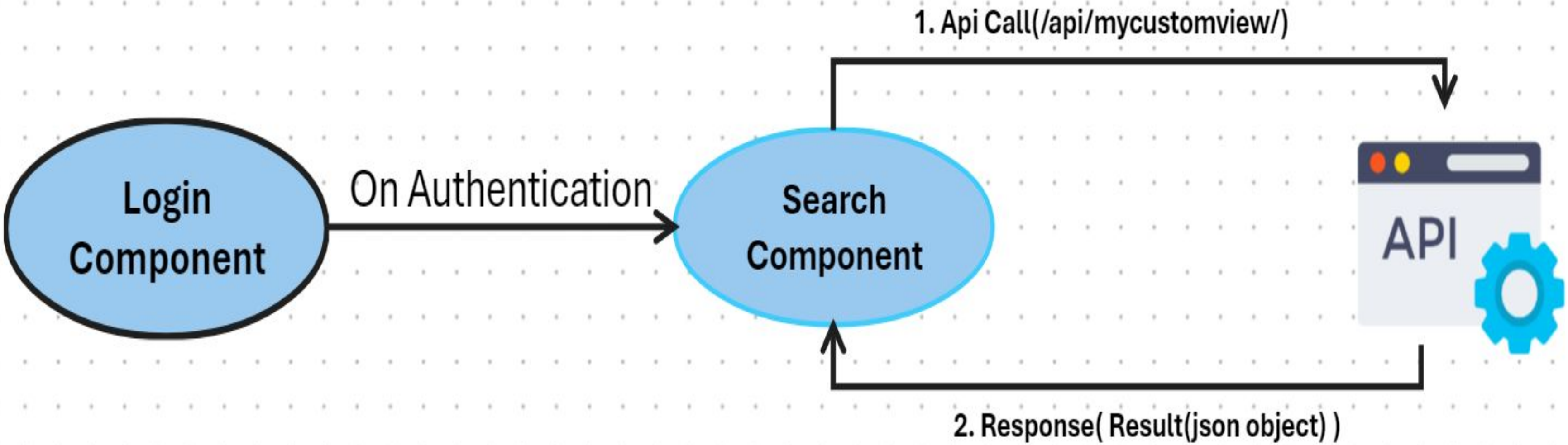
- Enter a valid book name and stream to perform a successful book search.
- Click the "Logout" button to log out successfully.

Login Component

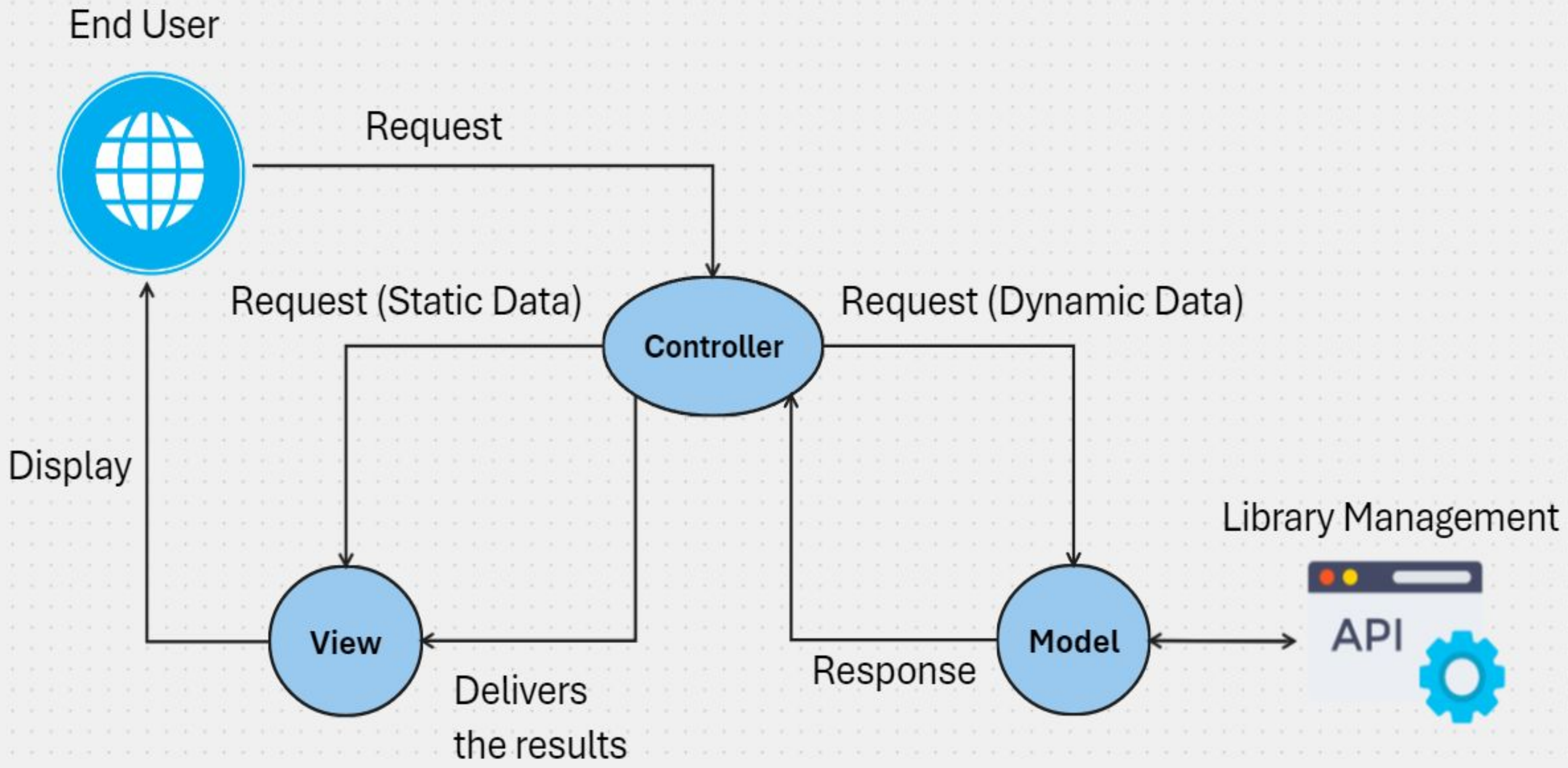


API
1. Authentication(api/api-token-auth/)
2. Response(Token)
Login Component
3. Enduser Auth(/api/end-user-auth/) with token
4.Response(Authenticated)
5. Redirect to Search Page
Search Component

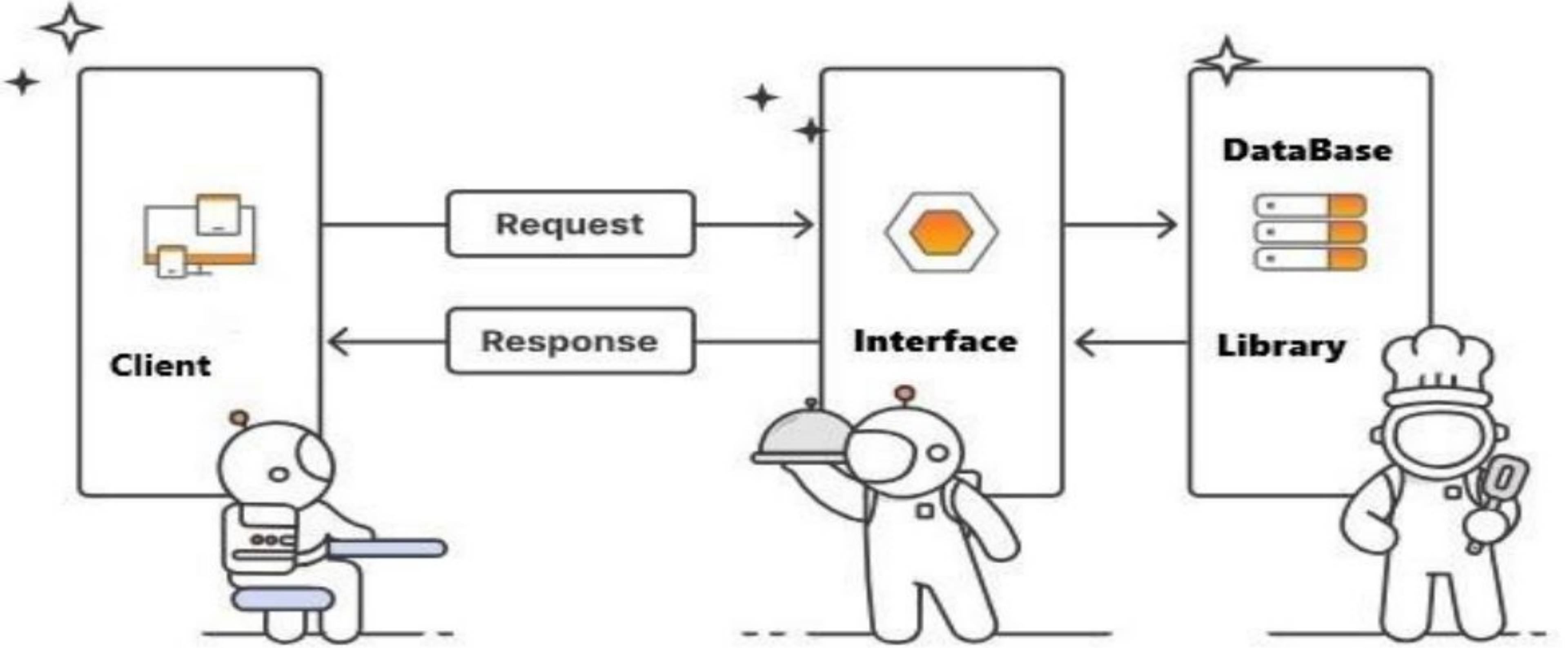
Search Component

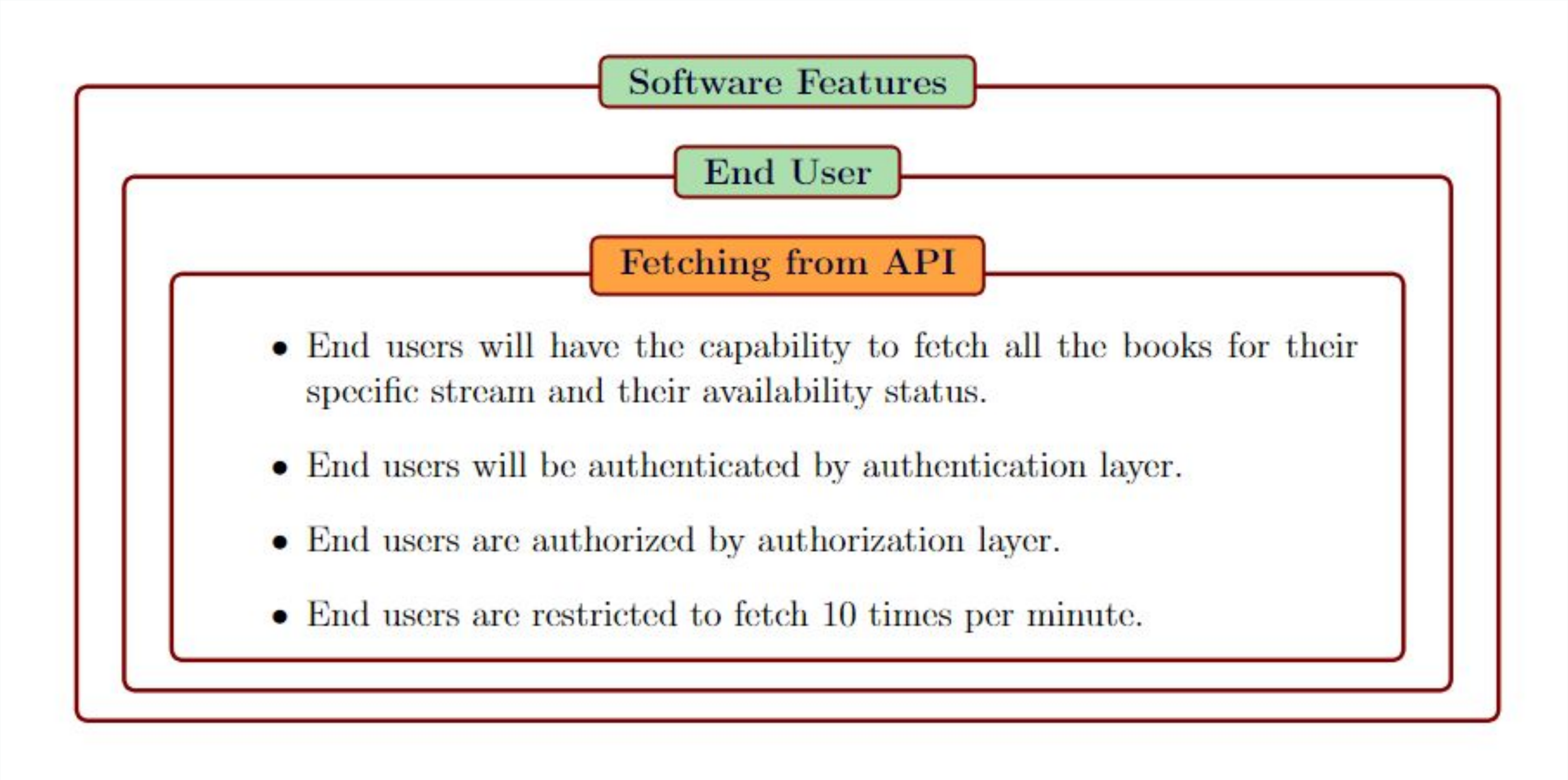

1. Api Call(/api/mycustomview/)
Login Component
On Authentication
Search Component
API
2. Response(Result(json object))

Software Architecture



Library Management Api





Software Features

End User

Fetching from API

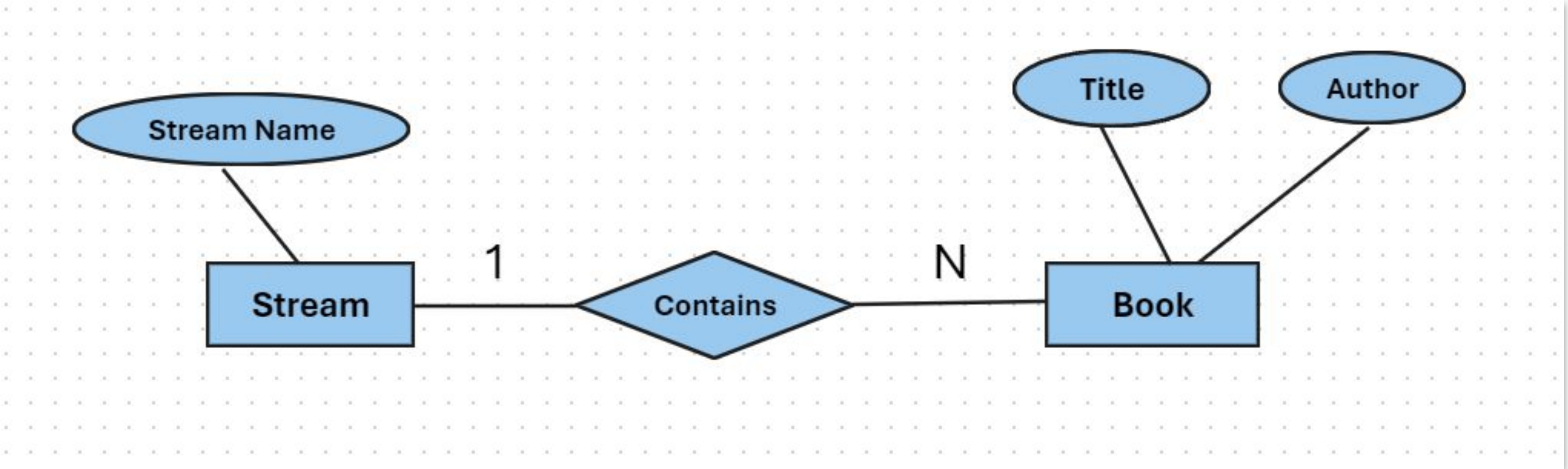
- End users will have the capability to fetch all the books for their specific stream and their availability status.
- End users will be authenticated by authentication layer.
- End users are authorized by authorization layer.
- End users are restricted to fetch 10 times per minute.

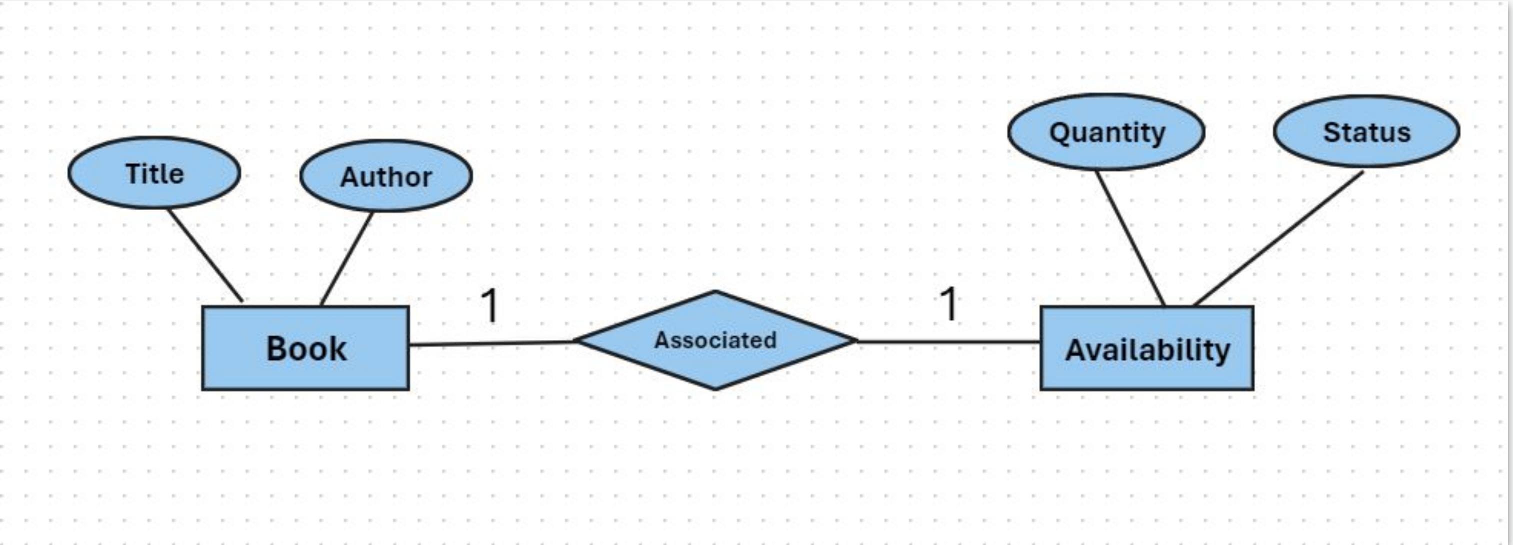
Software Features

Admin User

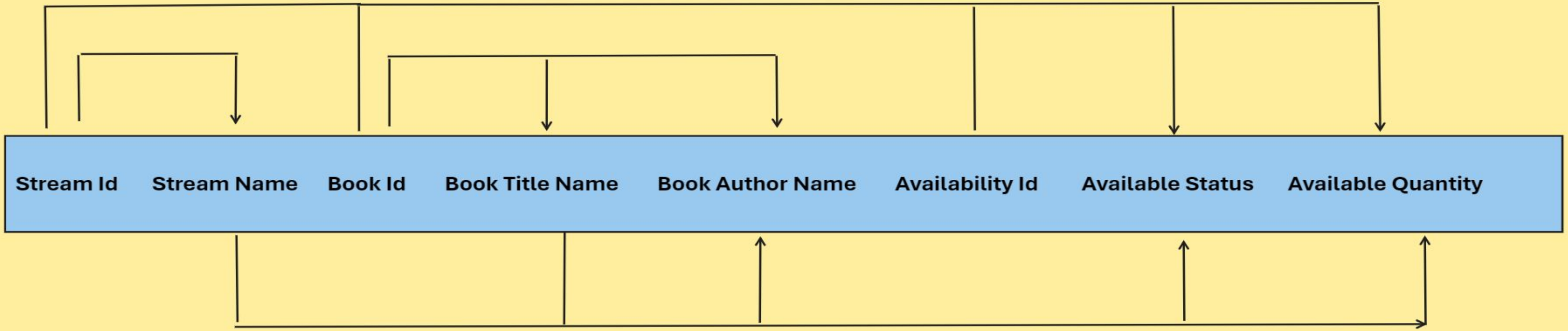
Performing CRUD on API

- Admin user will have capability to create, Read, update, and Delete the books, streams, and book availability status.
- Admin users are authenticated by authentication layer.
- Admin users are authorized by authorization layer.
- Admin users are restricted to perform operations 20 times per minute.


Stream Name
Title
Author
1
N
Stream
Contains
Book


Title
Author
Quantity
Status
1
1
Book
Associated
Availability

Functional Dependencies



[Stream Id] ⟹ [Stream Name]

[Book Id] ⟹ [Book Title Name, Book Author Name]

[Book Id, Availability Id] ⟹ [Availability Status, Availability Quantity]

[Stream Name, Book Title Name] ==⟹ [SN, BTN, BAN, AQ, AS]

Normalization

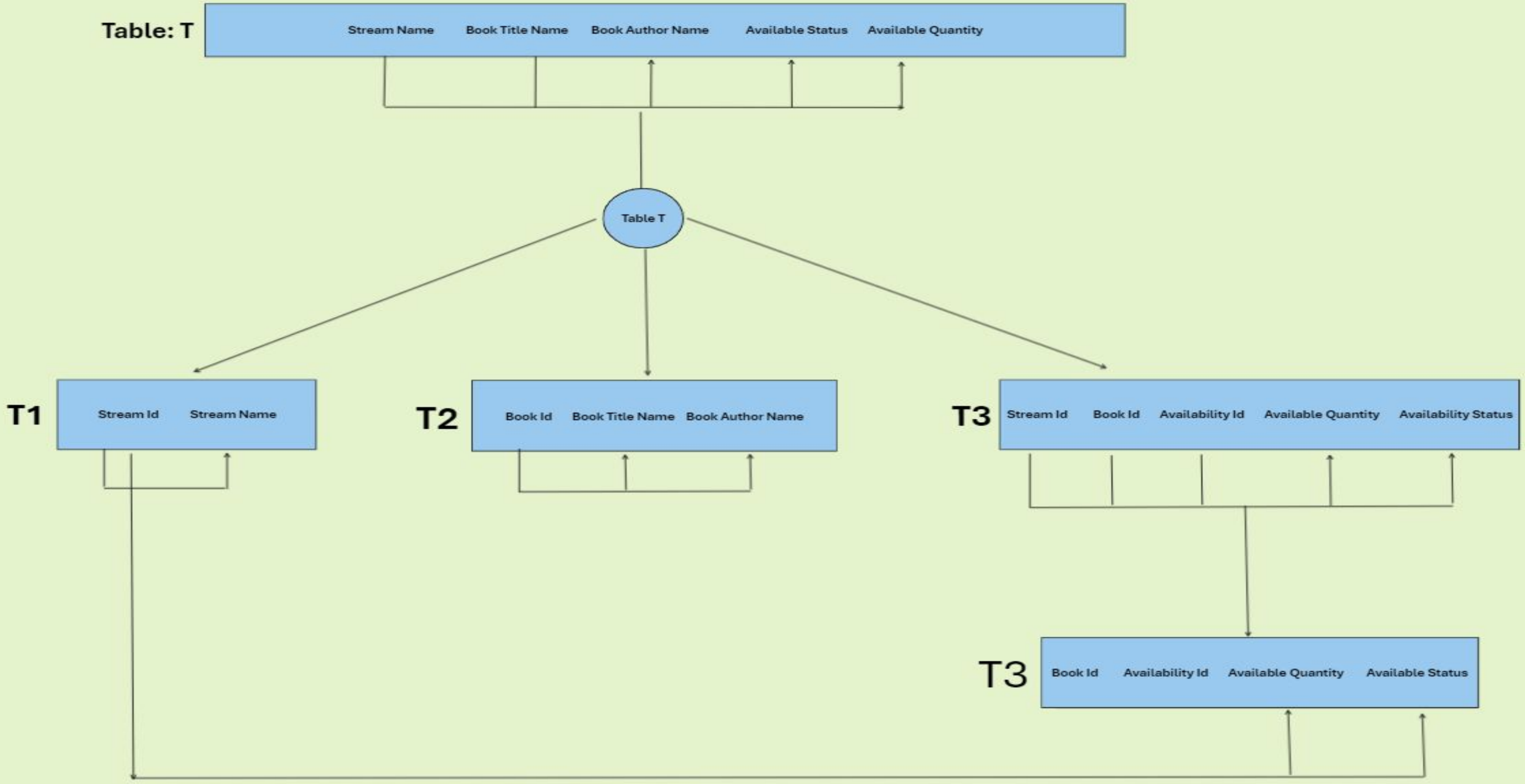


Table: T

Stream Name Book Title Name Book Author Name Available Status Available Quantity

Table T

T1

Stream Id Stream Name

T2

Book Id Book Title Name Book Author Name

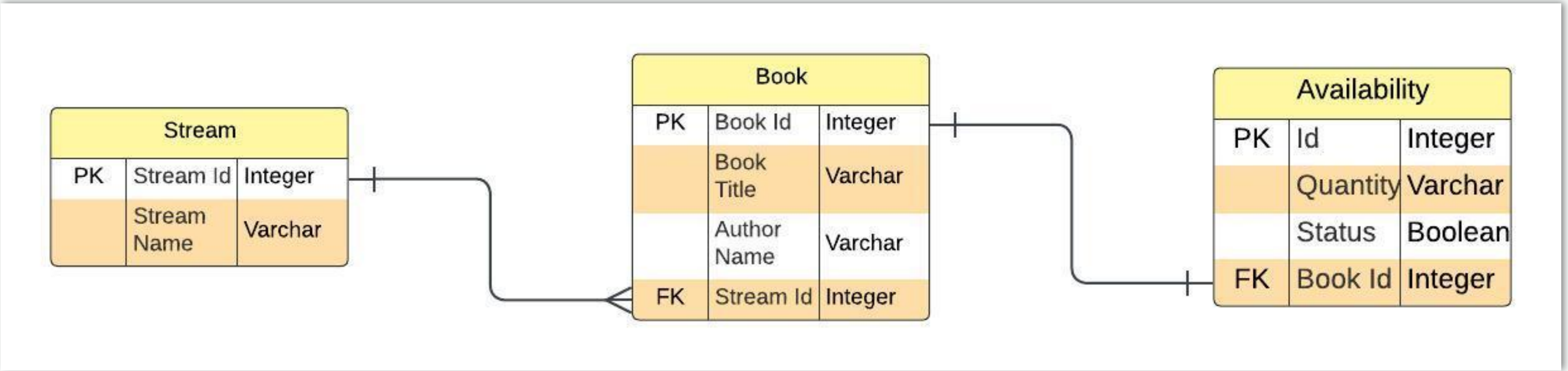
T3

Stream Id Book Id Availability Id Available Quantity Availability Status

T3

Book Id Availability Id Available Quantity Available Status

Crow’s Foot Notation



Stream

PK	Stream Id	Integer
	Stream Name	Varchar

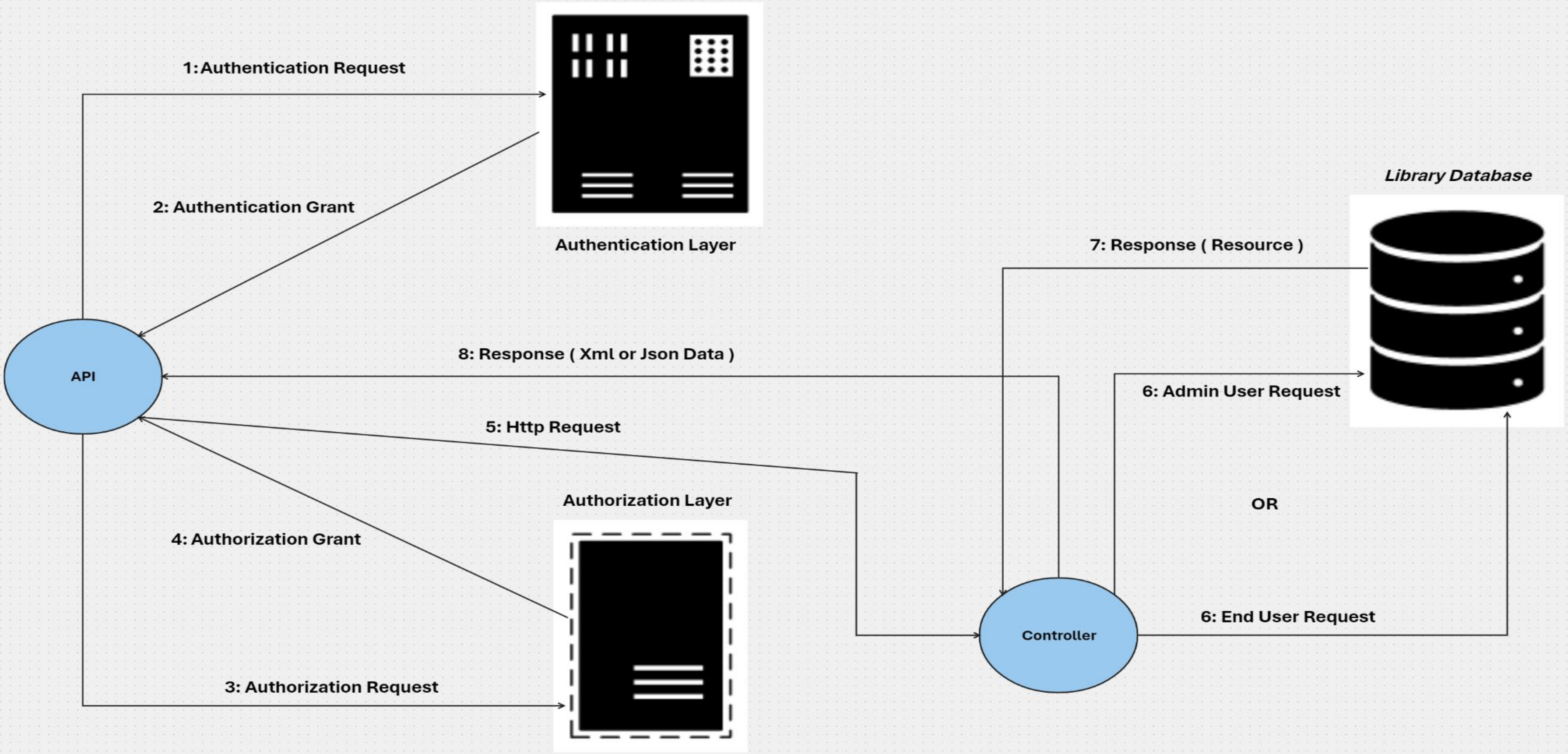
Book

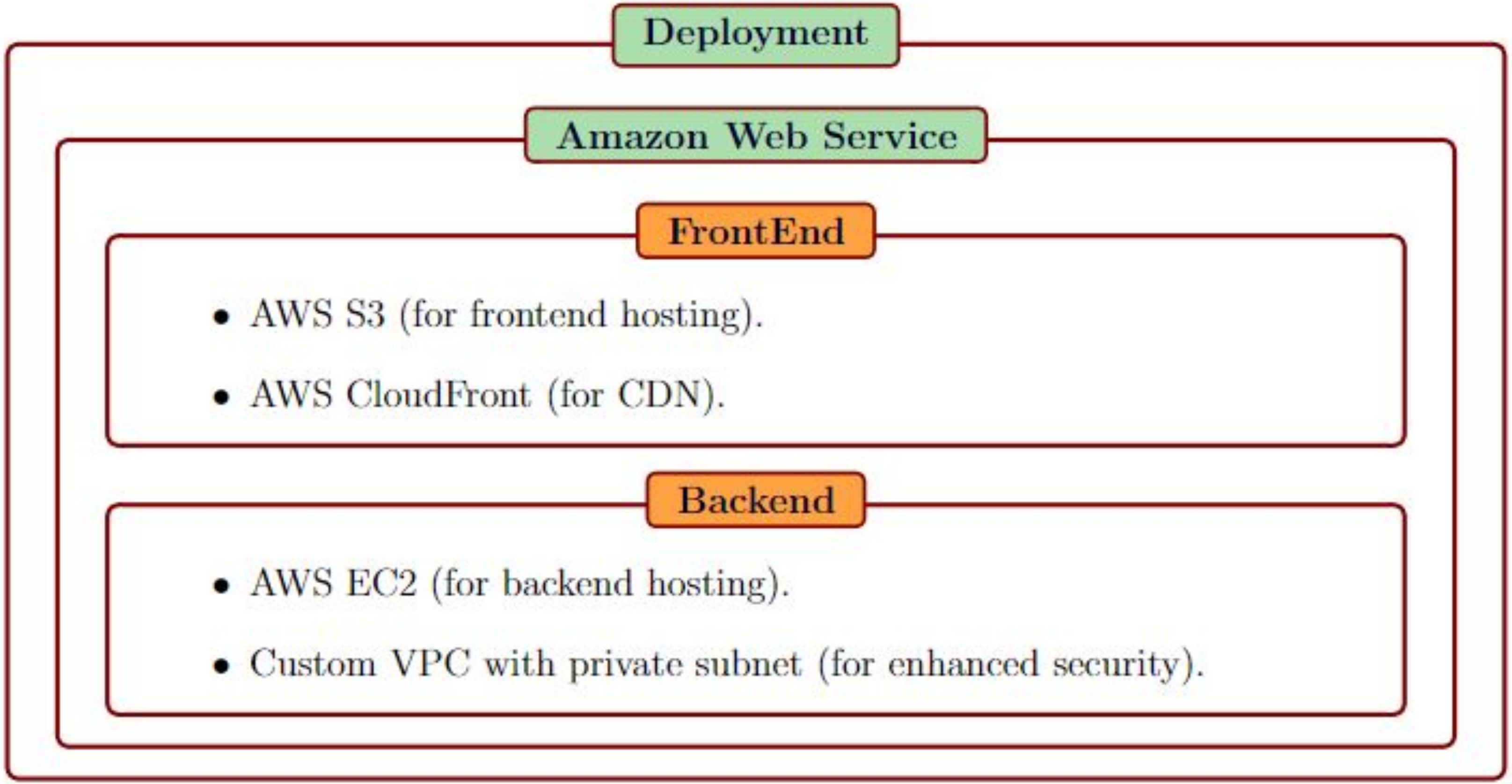
PK	Book Id	Integer
	Book Title	Varchar
	Author Name	Varchar
FK	Stream Id	Integer

Availability

PK	Id	Integer
	Quantity	Varchar
	Status	Boolean
FK	Book Id	Integer

Api Internal Architecture





Deployment

Amazon Web Service

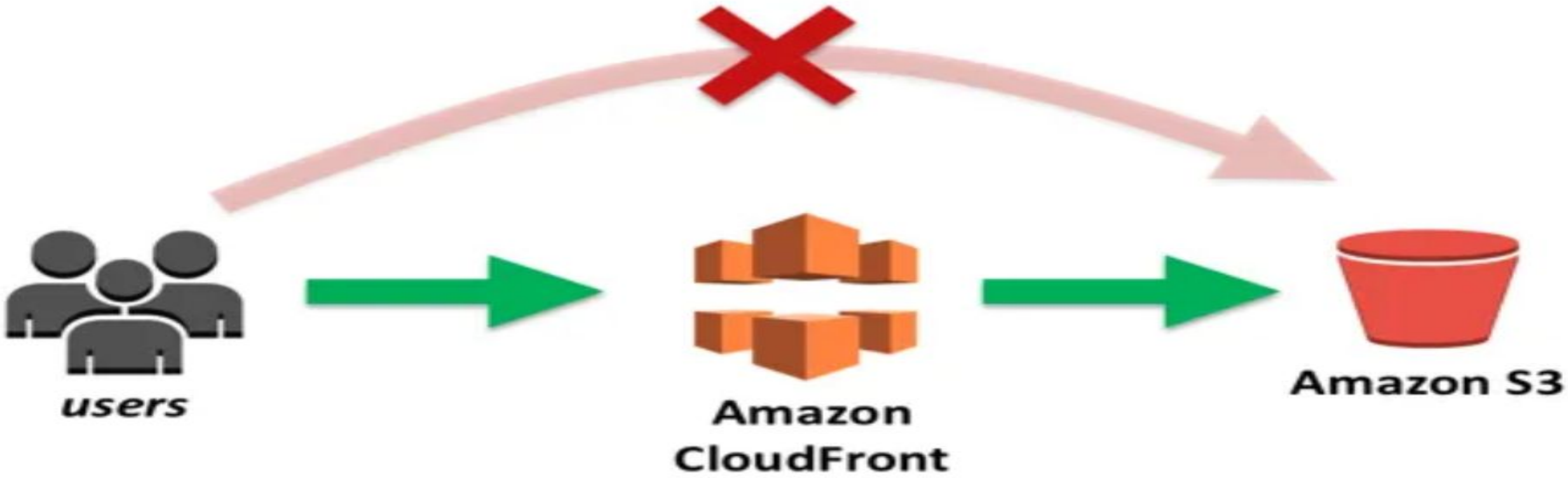
FrontEnd

- AWS S3 (for frontend hosting).
- AWS CloudFront (for CDN).

Backend

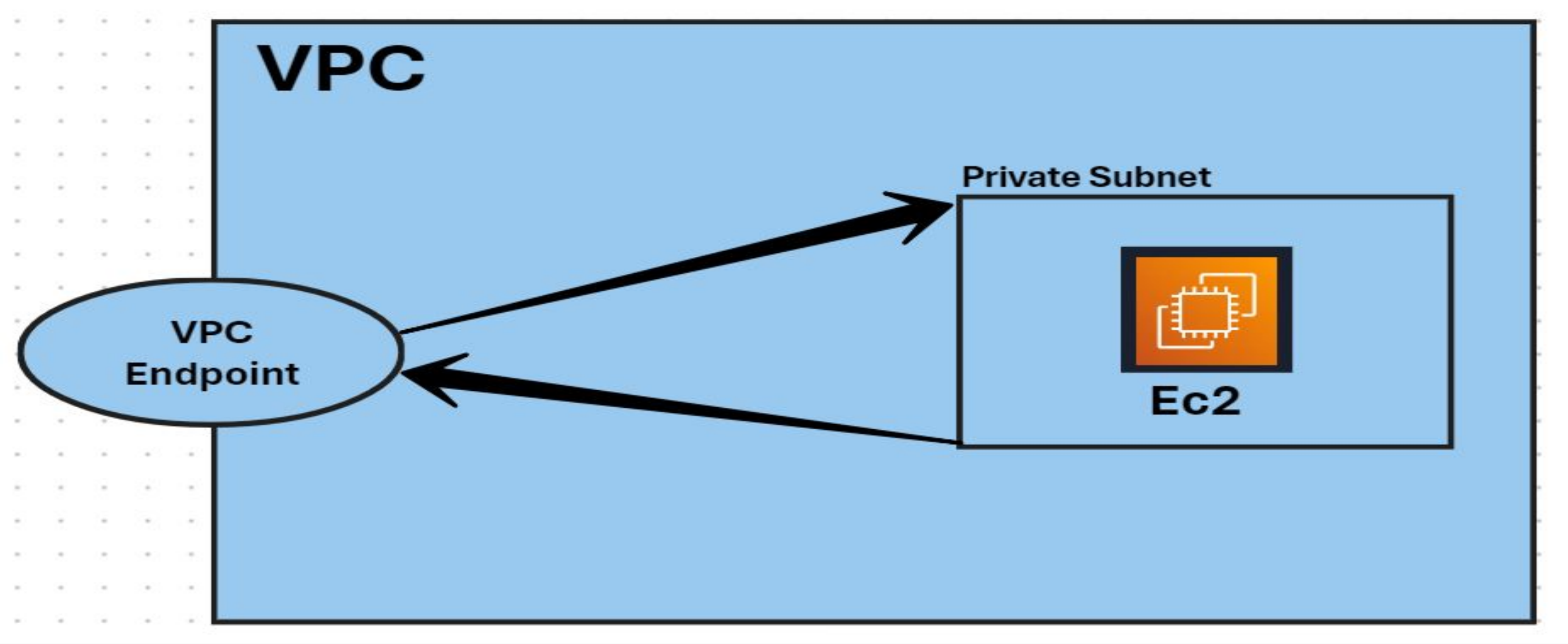
- AWS EC2 (for backend hosting).
- Custom VPC with private subnet (for enhanced security).

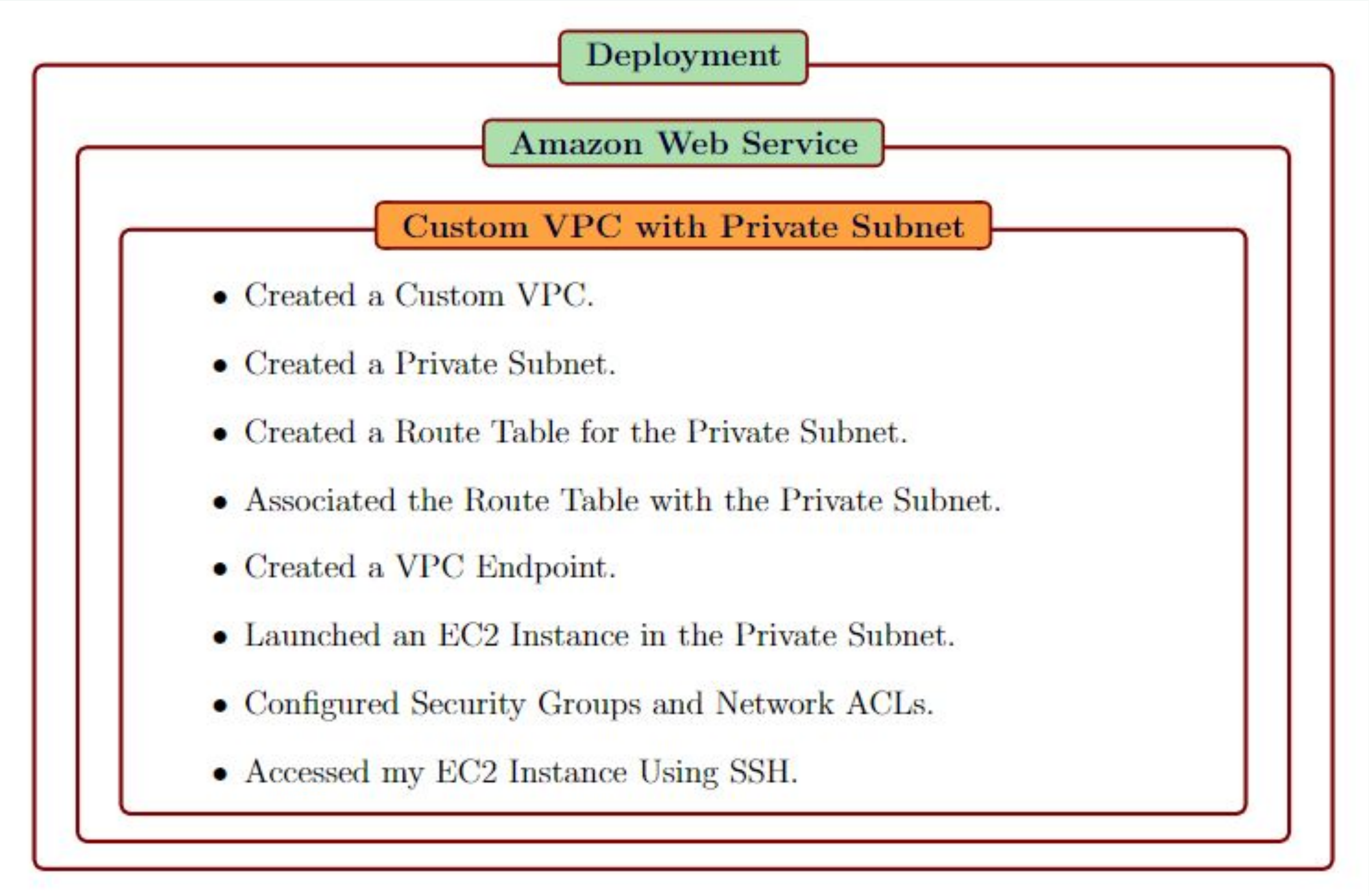
Amazon CloudFront



- Content delivery network service.
- Delivers our content through data centers called edge locations.
- Security, High Availability, Low Latency and High Transfer Speeds.

Custom VPC with private subnet



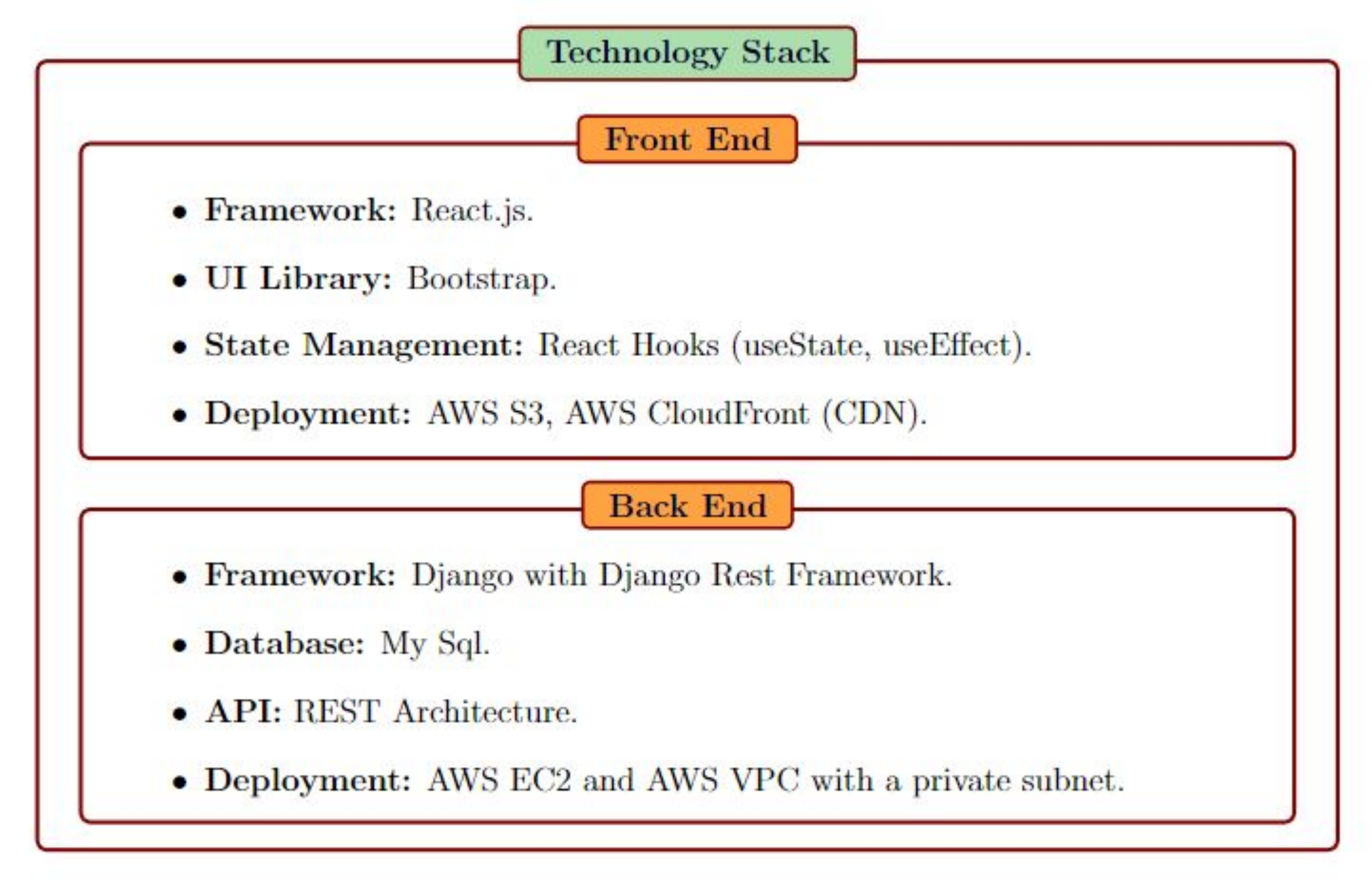


Deployment

Amazon Web Service

Custom VPC with Private Subnet

- Created a Custom VPC.
- Created a Private Subnet.
- Created a Route Table for the Private Subnet.
- Associated the Route Table with the Private Subnet.
- Created a VPC Endpoint.
- Launched an EC2 Instance in the Private Subnet.
- Configured Security Groups and Network ACLs.
- Accessed my EC2 Instance Using SSH.



Technology Stack

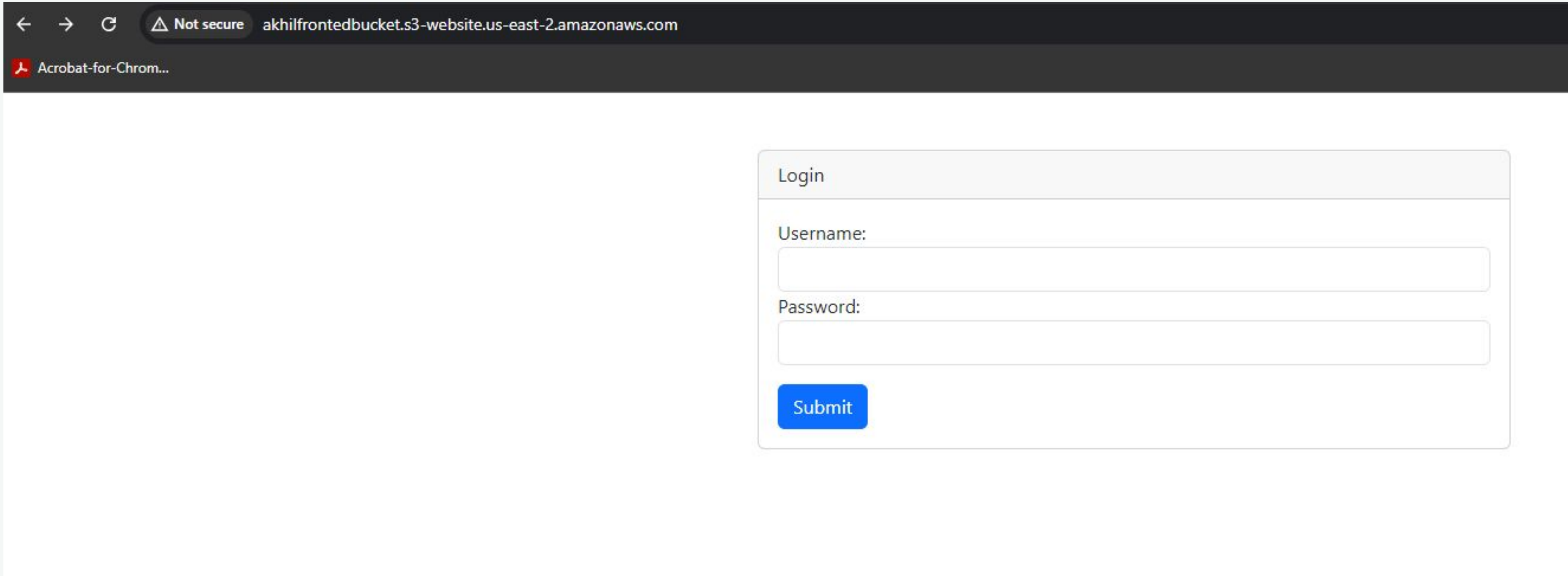
Front End

- **Framework:** React.js.
- **UI Library:** Bootstrap.
- **State Management:** React Hooks (useState, useEffect).
- **Deployment:** AWS S3, AWS CloudFront (CDN).

Back End

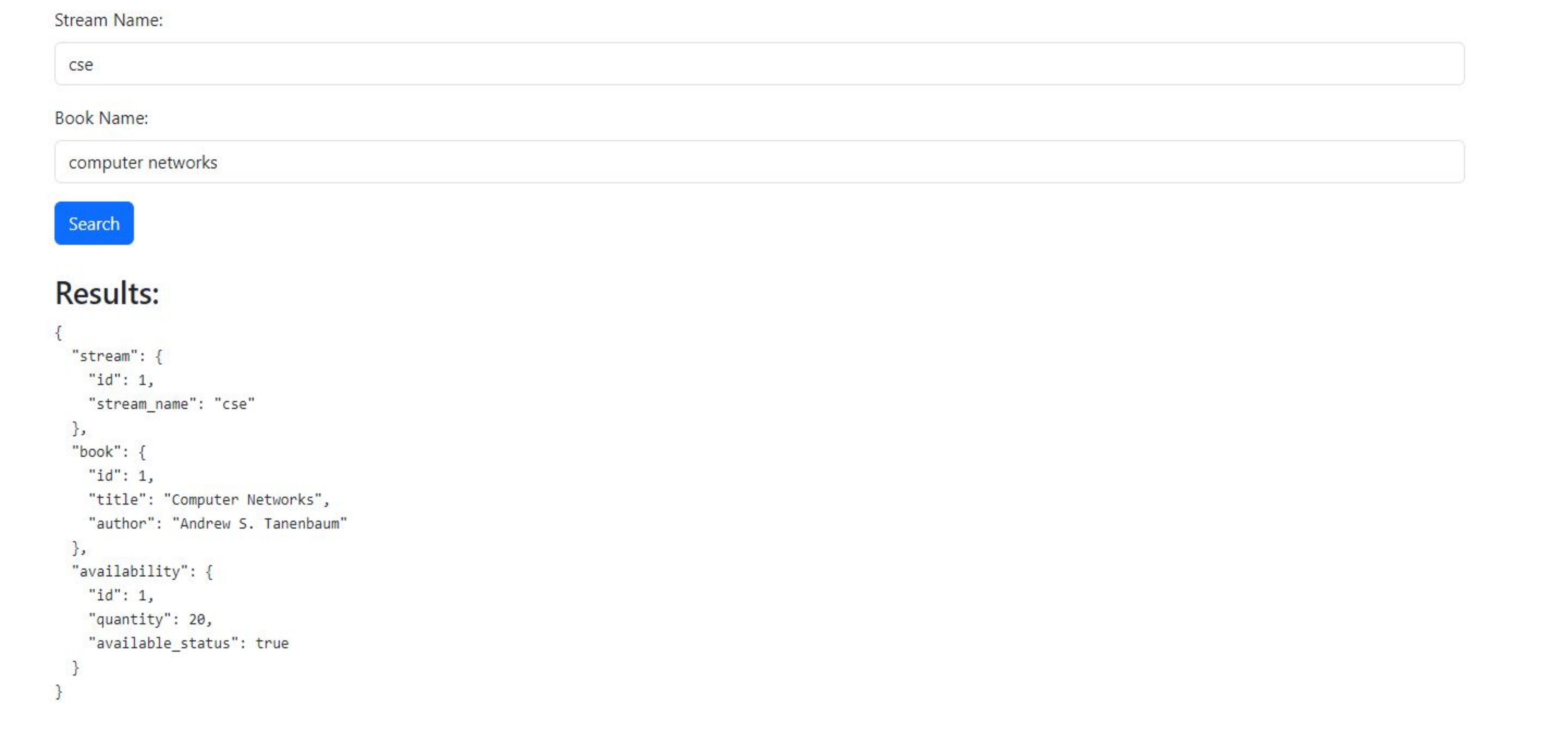
- **Framework:** Django with Django Rest Framework.
- **Database:** My Sql.
- **API:** REST Architecture.
- **Deployment:** AWS EC2 and AWS VPC with a private subnet.

Output



Not secure akhilfrontedbucket.s3-website.us-east-2.amazonaws.com

Output



Stream Name:

cse

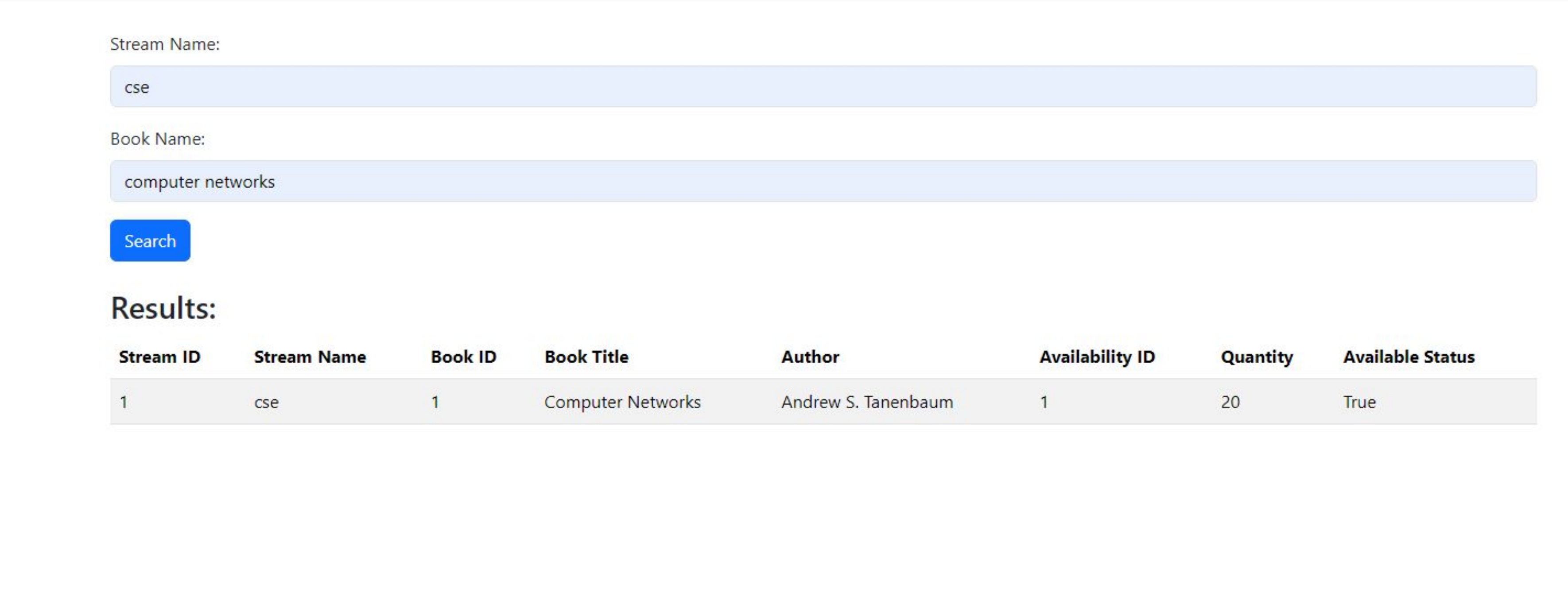
Book Name:

computer networks

Results:

```
{
  "stream": {
    "id": 1,
    "stream_name": "cse"
  },
  "book": {
    "id": 1,
    "title": "Computer Networks",
    "author": "Andrew S. Tanenbaum"
  },
  "availability": {
    "id": 1,
    "quantity": 20,
    "available_status": true
  }
}
```

Output



Stream Name: cse

Book Name: computer networks

Results:

Stream ID	Stream Name	Book ID	Book Title	Author	Availability ID	Quantity	Available Status
1	cse	1	Computer Networks	Andrew S. Tanenbaum	1	20	True