

CHALLENGE

TIM D - CHAPTER 5

RESERVASI TIKET BIOSKOP

(Database & SpringBoot)

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc108612506)

[Catatan Release Dokumen 4](#_Toc108612507)

[Version History 5](#_Toc108612508)

[I 6](#_Toc108612509)

[FLOW CHART AND ENTITY RELATIONSHIP DIAGRAM 6](#_Toc108612510)

[A. Project Flow 6](#_Toc108612511)

[B. Entity Relationship Diagram 7](#_Toc108612512)

[II. 8](#_Toc108612513)

[Tabel dan Insert Data 8](#_Toc108612514)

[1. TABLE 8](#_Toc108612515)

[2. INSERT DATA 13](#_Toc108612516)

[III 15](#_Toc108612517)

[RUNNING API 15](#_Toc108612518)

[.1. USER 16](#_Toc108612519)

[1.1 GET By ID 16](#_Toc108612520)

[1.2 PUT 17](#_Toc108612521)

[1.3 DELETE 20](#_Toc108612522)

[1.4 POST 22](#_Toc108612523)

[1.5 GET ALL 24](#_Toc108612524)

[2. Seats 27](#_Toc108612525)

[2.1 GET By ID 27](#_Toc108612526)

[2.2 PUT 28](#_Toc108612527)

[2.3 DELETE 30](#_Toc108612528)

[2.4 GET ALL 32](#_Toc108612529)

[2.5 POST 35](#_Toc108612530)

[2.6 POST IS AVAILABLE 37](#_Toc108612531)

[3. SCHEDULE 39](#_Toc108612532)

[3.1 GET ALL 39](#_Toc108612533)

[3.2 PUT SCHEDULE 41](#_Toc108612534)

[3.3 POST/CREATE SCHEDULE 44](#_Toc108612535)

[3.4 POST/FIND SCHEDULE BY FILM NAME 49](#_Toc108612536)

[3.5 POST/ FIND SCHEDULE BY FILM NAME LIKE 50](#_Toc108612537)

[3.6 GET BY ID 52](#_Toc108612538)

[3.7 DELETE 54](#_Toc108612539)

[55](#_Toc108612540)

[4. FILM 56](#_Toc108612541)

[4.1 GET BY ID 56](#_Toc108612542)

[4.2 PUT 58](#_Toc108612543)

[4.3 DELETE 59](#_Toc108612544)

[4.4 GET ALL 61](#_Toc108612545)

[4.5 POST 63](#_Toc108612546)

[4.6 POST FILM IS PLAYING 66](#_Toc108612547)

[5. BOOKING 68](#_Toc108612548)

[5.1 GET ALL 68](#_Toc108612549)

[5.2 GET BY ID 71](#_Toc108612550)

[5.3 CREATE/POST 73](#_Toc108612551)

[5.4 UPDATE/PUT 76](#_Toc108612552)

[5.5 DELETE 80](#_Toc108612553)

[5.6 POST/QUERY BY FILMNAME 81](#_Toc108612554)

[5.7 Cetak Tiket Reservasi Bioskop 83](#_Toc108612555)

# Catatan Release Dokumen

|  |  |  |  |
| --- | --- | --- | --- |
| **Tanggal Release 20 Juni**  **2022** | **Versi** | **Oleh** | **Catatan**  **Perubahan** |
| 1.0.0 | Tim C CH 4 | Versi Pertama |
| **20 Juni**  **2022** | 1.0.1 | Tim C CH 4 | Revisi Pertama |
| **21 Juni**  **2022** | 1.0.2 | Tim C CH 4 | Versi Kedua |
| **28 Juni**  **2022** | 1.0.3 | TIM C CH 4 | Versi Ketiga |
| **30 Juni 2022** | 1.0.4 | TIM D CH 5 | Versi Keempat |
| **11 Juli 2022** | 1.05 | TIM D CH 5 | Versi Kelima |

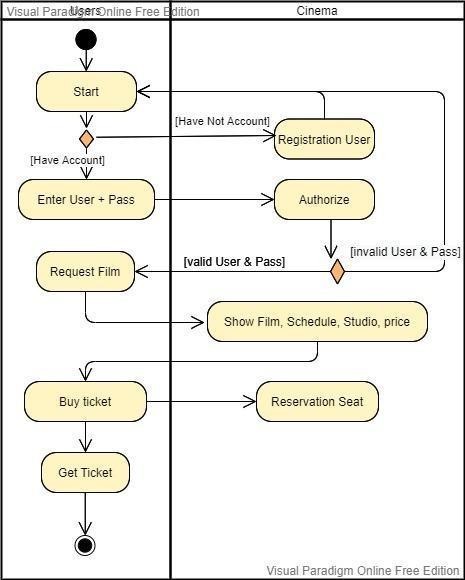
# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| DATE | VERSION | AUTHOR | DESCRIPTION |
| 20/06/2022 | 1.0.0 | TIM C CH 4 | BERISI PROJECT FLOW, ERD, DAN QUERY PEMBUATAN TABEL  (Implementasi query DDL) DAN MELAKUKAN SELECT, INSERT, DAN  JOIN  (Implementasi Query DML) PADA DATABASE  BIOSKOP DB |
| 28/06/2022 | 1.0.1 | TIM C CH 4 | MENAMBAHKAN  RESPON HANDLER DI SETIAP ENDPOINT |
| 30/06/2022 | 1.0.2 | TIM D CH 5 | MIGRASI MYSQL KE POSTGREE |
| 01/07/2022 | 1.0.3 | TIM D CH 5 | SETUP TRELLO |
| 02/07/2022 | 1.0.4 | TIM D CH 5 | DDL DAN DML |
| 04/07/2022 | 1.0.5 | TIM D CH 5 | MEMBUAT THYMELEAF |
| 11/07/2022 | 1.0.6 | TIM D CH 5 | FRONT END CRUD |

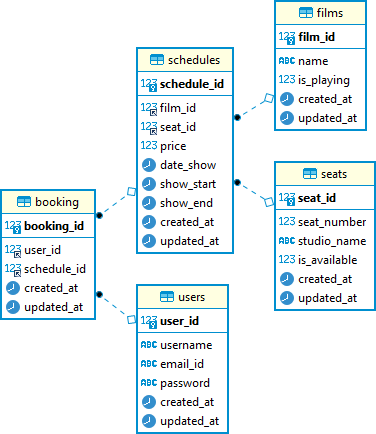
# I

# FLOW CHART AND ENTITY RELATIONSHIP DIAGRAM

## Project Flow



## Entity Relationship Diagram



II.

Tabel dan Insert Data

1. TABLE

* Users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Not Null | Auto Increment | Key |
| *user\_id* | int | - | v | - | Primary Key |
| *username* | varchar | 50 | - | - | - |
| *email\_id* | varchar | 255 | - | - | - |
| *password* | varchar | 255 | - | - | - |
| *created\_at* | timesta mp | - | v | - | - |
| *updated\_at* | datetime | - | v | - | - |

Query :

**create table** Users(

user\_id **int SERIAL primary key not null**,

username **varchar**(50) **null**,

email\_id **varchar**(255) **null**,

“**password” varchar**(255) **null**,

created\_at TIMESTAMP **DEFAULT CURRENT\_TIMESTAMP**

update\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

updated\_at TIMESTIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TRIGGER set\_timestamp

BEFORE UPDATE ON users

FOR EACH ROW

EXECUTE PROCEDURE trigger\_set\_timestamp();

* films

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Not Null | Auto Increment | Key |
| *film\_id* | int |  | v | - | Primary Key |
| *name* | varchar | 100 | v | - | - |
| *is\_playing* | tinyint | - | v | - | - |
| *Created\_at* | timestamp | - | v | - | - |
| *Updated\_at* | datetime | - | v | - | - |

Query :

**create table** films(

film\_id **SERIAL NOT null primary key,**

name **varchar**(100) **not null**,

is\_playing **int not null**,

created\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP** ,

updated\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

);

CREATE TRIGGER set\_timestamp

BEFORE UPDATE ON films

FOR EACH ROW

EXECUTE PROCEDURE trigger\_set\_timestamp();

* schedules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Not Null | Auto Increment | Key |
| *schedule\_id* | int | - | v | - | Primary Key |
| *film\_id* | int | - | v | - | Foreign Key |
| *date\_show* | date | - | v | - | - |
| *show\_start* | time | - | v | - | - |
| *show\_end* | time | - | v | - | - |
| *price* | int | - | v | - | - |
| *created\_at* | timesta mp | - | v | - | - |
| *updated\_at* | datetime | - | v | - | - |

Query :

**create table** schedules**(**

schedule\_id **SERIAL primary key not null**

film\_id **int not null,** seat\_id **int not null,** price **int not null,**

date\_show **date not null,** show\_start **time not null,** show\_end **time not null,**

created\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ,**

updated\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**foreign key (film\_id) REFERENCES Films(film\_id), foreign key (seat\_id) REFERENCES Seats(seat\_id)**

**);**

**CREATE TRIGGER set\_timestamp**

**BEFORE UPDATE ON schedules**

**FOR EACH ROW**

**EXCECUTE PROCEDURE trigger\_set\_timestamp();**

* seats

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Not Null | Auto Increment | Key |
| *seat\_id* | int | - | v | - | Primary Key |
| *seat\_numbe r* | int | - | v | - | - |
| *studio\_nam e* | varchar | 50 | v | - | - |
| *is\_available* | tinyint | - | v | - | - |
| *created\_at* | timestam p | - | v | - | - |
| *updated\_at* | datetime | - | v | - | - |

Query :

**create table** Seats(

seat\_id **SERIAL primary key not null,**

seat\_number **int not null**,

studio\_name **varchar**(50) **not null**, is\_available **boolean not null default** 0,

created\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP** ,

updated\_at **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

);

CREATE TRIGGER set\_timestamp

BEFORE UPDATE ON seats

FOR EACH ROW

EXECUTE PROCEDURE trigger\_set\_timestamp();

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Length | Not Null | Auto Increment | Key |
| *booking\_id* | int | - | v | - | Primary Key |
| *user\_id* | int | - | v | - | Foreign Key |
| *schedule\_id* | int | - | v | - | Foreign Key |
| *created\_at* | timestam p | - | v | - | - |
| *updated\_at* | datetime | - | v | - | - |

* booking

Query :

**create table Booking(**

**booking\_id SERIAL primary key not null,**

**user\_id int,**

**schedule\_id int,**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**foreign key (user\_id) references Users(user\_id),**

**foreign key (schedule\_id) references Schedules(schedule\_id)**

**);**

**CREATE TRIGGER set\_timestamp**

**BEFORE UPDATE ON booking**

**FOR EACH ROW**

**EXECUTE PROCEDURE trigger\_set\_timestamp();**

**CREATE OR REPLACE FUNCTION trigger\_set\_timestamp()**

**RETURN TRIGGER AS $$**

**BEGIN**

**NEW.updated\_at = NOW();**

**END;**

**$$ LANGUAGE plpsql**

1. INSERT DATA
   * Insert data users table

**INSERT INTO** users (username,email\_id,**password**)

**VALUES**

('value','value','value');

Example:

**INSERT INTO** users (username,email\_id,**password**)

**VALUES**

('Wahyu Wijanarko','WahyuW@gmail.com','Rahasiaaaa4x'), ('Sri Mulyani','S\_Mulyani@gmail.com','Inip45sword');

* + Insert data films table

**INSERT INTO** films (name,is\_playing)

**VALUES**

('value','value');

Example:

**INSERT INTO** films (name,is\_playing)

**VALUES**

('Jurassic World Dominion',1), ('The Rondup',1);

* + Insert data schedules table

**INSERT INTO** schedules (film\_id,date\_show,show\_start,show\_end,price,seat\_id)

**VALUES**

('value','value','value','value','value',’value’);

Example:

**INSERT INTO** schedules (film\_id,date\_show,show\_start,show\_end,price,seat\_id)

**VALUES**

(1,'2022-06-19','2022-06-30 18:00:00','2022-06-30 19:00:00',35000,1),

(2,'2022-06-19','2022-06-30 18:00:00','2022-06-30 19:00:00',35000,2);

* + Insert data seats table

**INSERT INTO** seats (seat\_number,studio\_name,is\_available)

**VALUES**

('value','value','value');

Example:

**INSERT INTO** seats (seat\_number,studio\_name,is\_available)

**VALUES**

(1,'A',1),

(2,'A',1);

* + Insert data booking table

**INSERT INTO** booking (user\_id, schedule\_id)

**VALUES**

('value','value');

Example:

**INSERT INTO** booking (user\_id, schedule\_id)

**VALUES**

(6,3),

(4,4)

# III

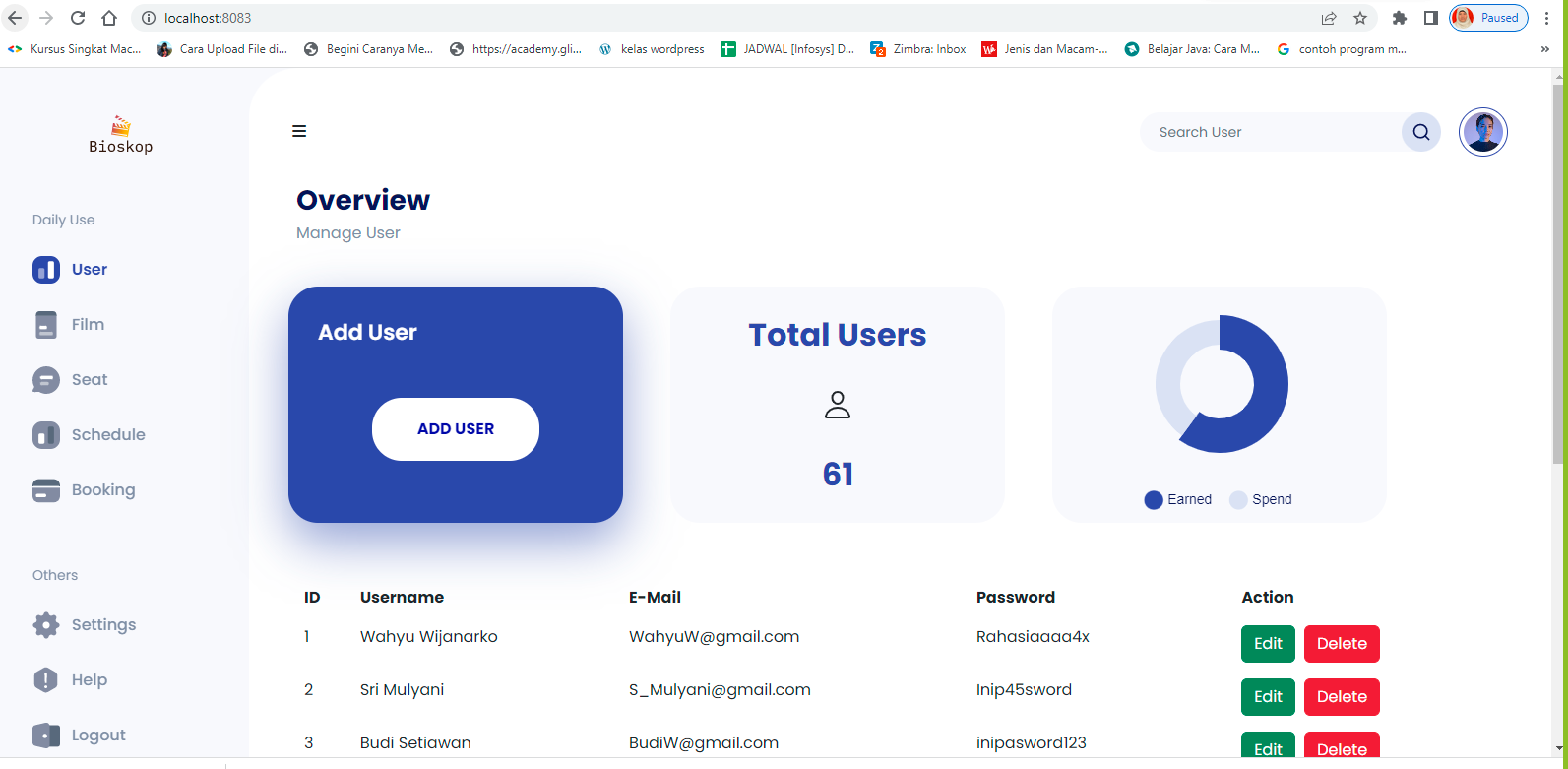
# RUNNING API

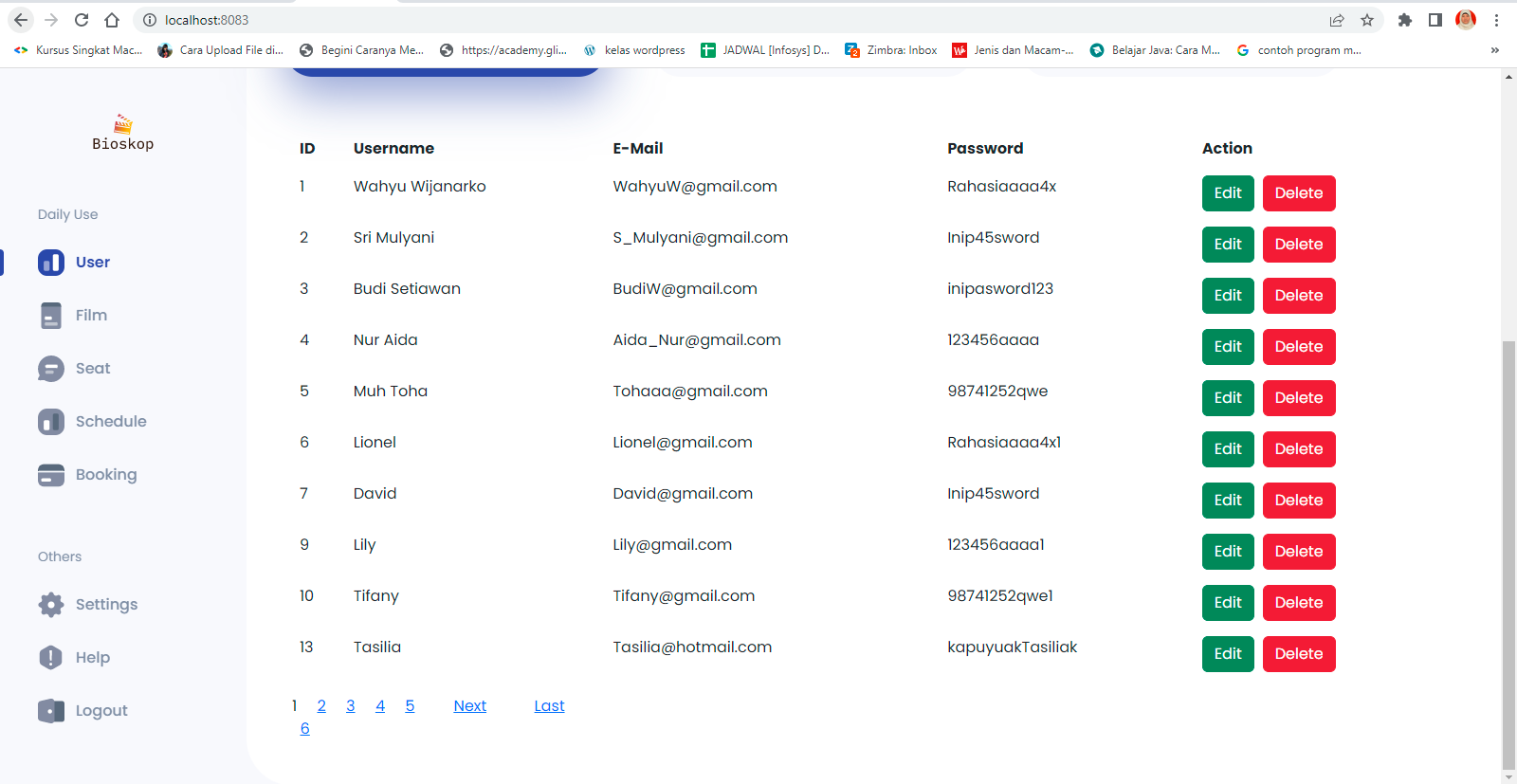
* Ringkasan

Semua permintaan BIOSKOP API untuk challenge chapter 5

* Resources
* Endpoint
* Home

Halaman Utama dari sebuah Bioskop-Api





* End Point Front End :

<http://localhost:8083>

* 1. USER
  2. GET By ID
* End Point swagger:

<http://localhost:8083/swagger-ui/index.html#/user-controller>

● Request URL

http://localhost:8083/swagger-ui/index.html#/user-controller

* Sample Request

{

“userId” : 6

}

* Sample Response Request

|  |  |  |
| --- | --- | --- |
| Success Response | | |
| "data" [ |  |  |
| "ID | " | 6 |
| "Username | " | "Lionel" |
| "Email | " | ["Lionel@gmail.com"](mailto:Lionel@gmail.com) |
| "Password | " | "Rahasiaaaa4x1" |

}

"message" "Successfully Get User By ID!" "status" 200

}

}

"data": "Data Not Found!",

"message": "User not exist with id :66666", "status": 404

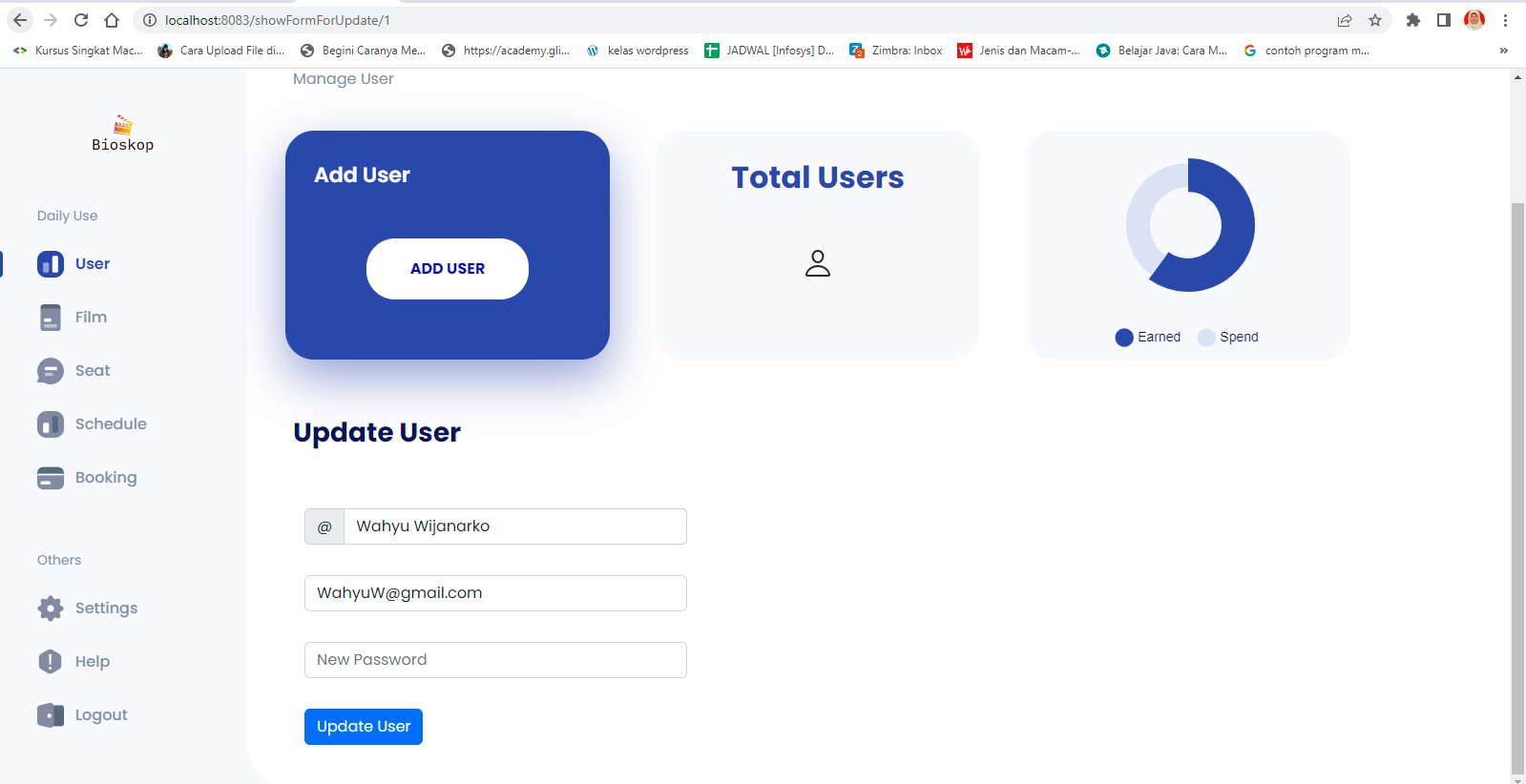
{

Failed Response

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name yang dimiliki oleh  konsumen. | Lily |
| emailId | String | M |  | Email yang digunakan oleh  konsumen saat mendaftar. | Lily@gma  il.com |
| password | String | M |  | Password yang diinput oleh  konsumen saat mendaftar. | 123456aaa  a1 |
| userId | Long | M |  | User Id yang diberikan system  untuk konsumen. | 9 |

* 1. PUT
* Tampilan Update Users digunakan untuk merubah data user baru yang berdasarkan username, email dan password.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/user-controller/updateUser>

* End Point Front End:

http://localhost:8083/showFormForUpdate/1

* Request URL

http://localhost:8083/swagger-ui/index.html#/user-controller/updateUser

* Sample Request

{

"userId": 8,

"username": "string", "emailId": "string",

"password": "string"

}

* Sample Response Request

|  |  |
| --- | --- |
| Success Response | |
| { |  |
|  | "data" |
|  | "userId" 8 |
|  | "username" "string" |
|  | "emailId" "string" |
|  | "password" "string" |
|  | "message" "Successfully Updated User!" |
|  | "status" 200 |
| } |  |

}

"data": "Data Not Found!",

"message": "User not exist with id :66666", "status": 404

{

Failed Response

* + Request Structure

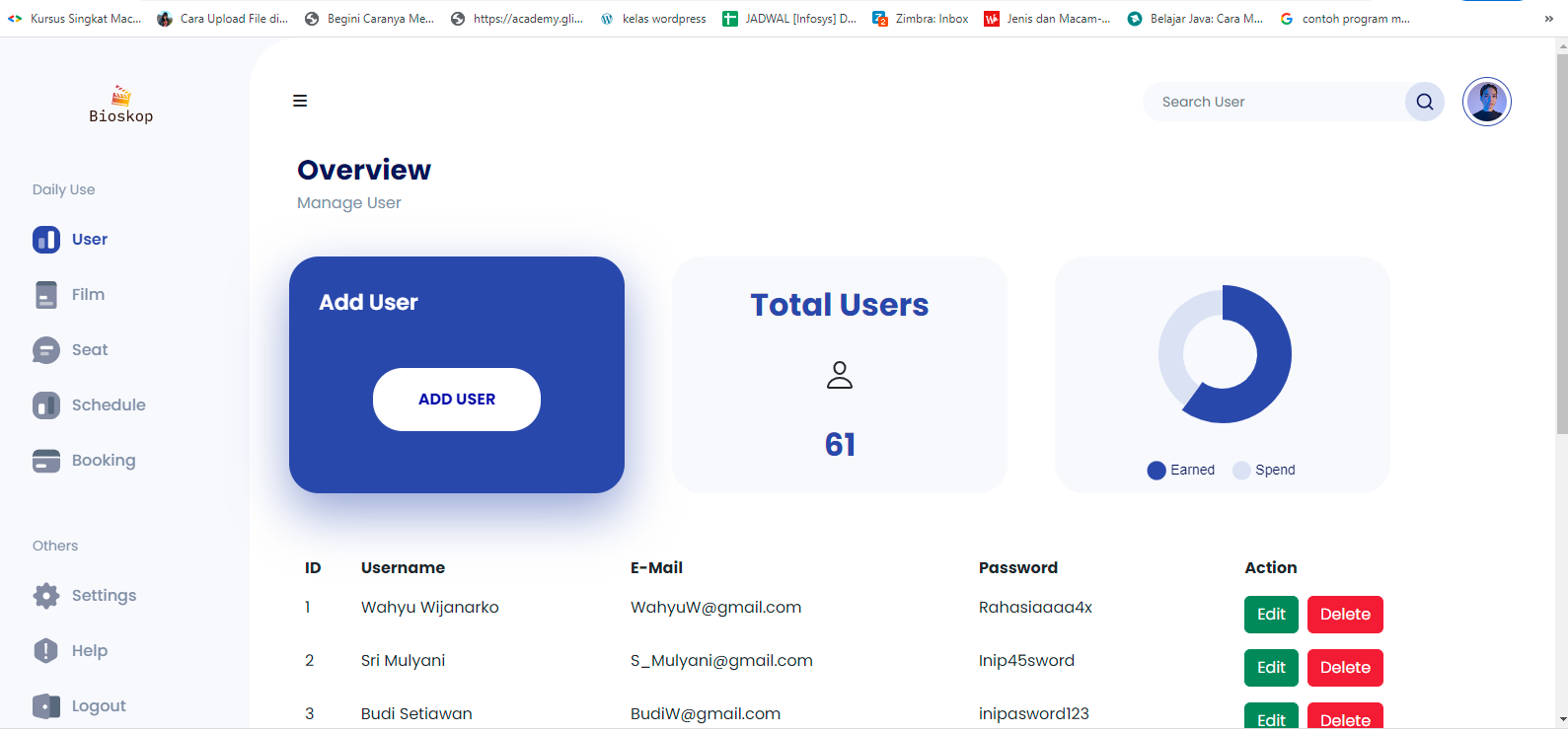
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name baru konsumen  yang ingin diupdate. | Lily |
| emailId | String | M |  | Email baru konsumen yang  ingin diupdate. | Lily@gm  ail.com |
| password | String | M |  | Password baru konsumen  yang ingin diupdate. | 123456aa  aa1 |
| userId | Long | M |  | User Id konsumen yang ingin | 9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | diupdate.. |  |

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name baru yang telah  diupdate. | Lily |
| emailId | String | M |  | Email baru baru yang telah  diupdate. | Lily@gm  ail.com |
| password | String | M |  | Password baru baru yang telah  diupdate. | 123456aa  aa1 |
| userId | Long | M |  | User Id konsumen yang ingin  diupdate.. | 9 |

* 1. DELETE
* Tampilan delete untuk menghapus data berdasarkan ID, Username, Email, password.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/deleteseats>

* End Point Front End :

http://localhost:8083/usersPage

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/deleteseats

* End Point Front End:

http://localhost:8083/usersPage

* Sample Request

{

userId : 8

}

* Sample Response Request

"data": {

{

Success Response

"deleted": {

**“headers”:{}**

**“body”:{**

**“deleted : true**

},

“statusCodeValue”: 200,

"message": " Delete User! ",

"status": 200

}

}

"data": "Data Not Found!",

"message": "User not exist with id :8999999", "status": 404

{

Failed Response

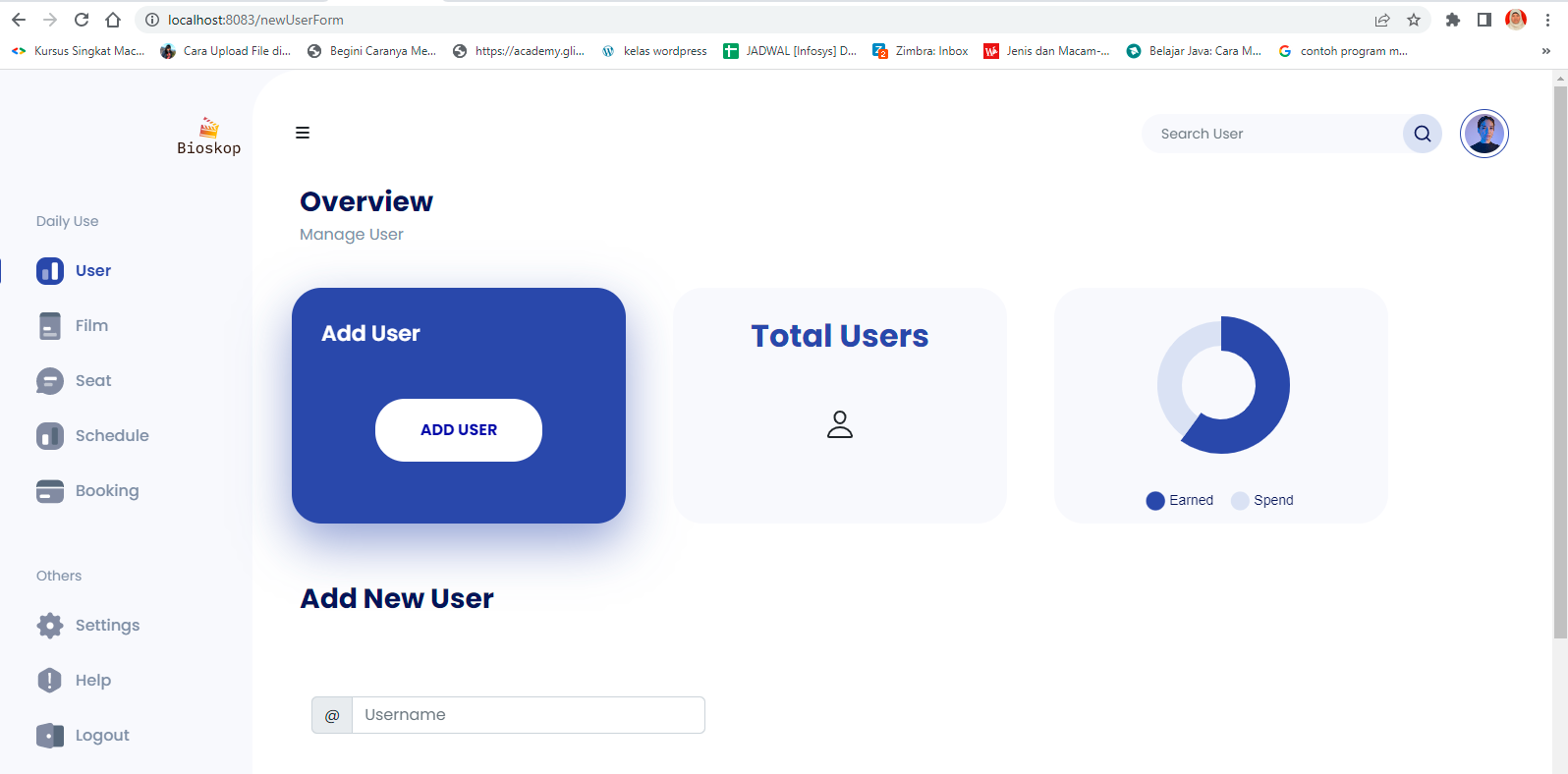
* Request Structure

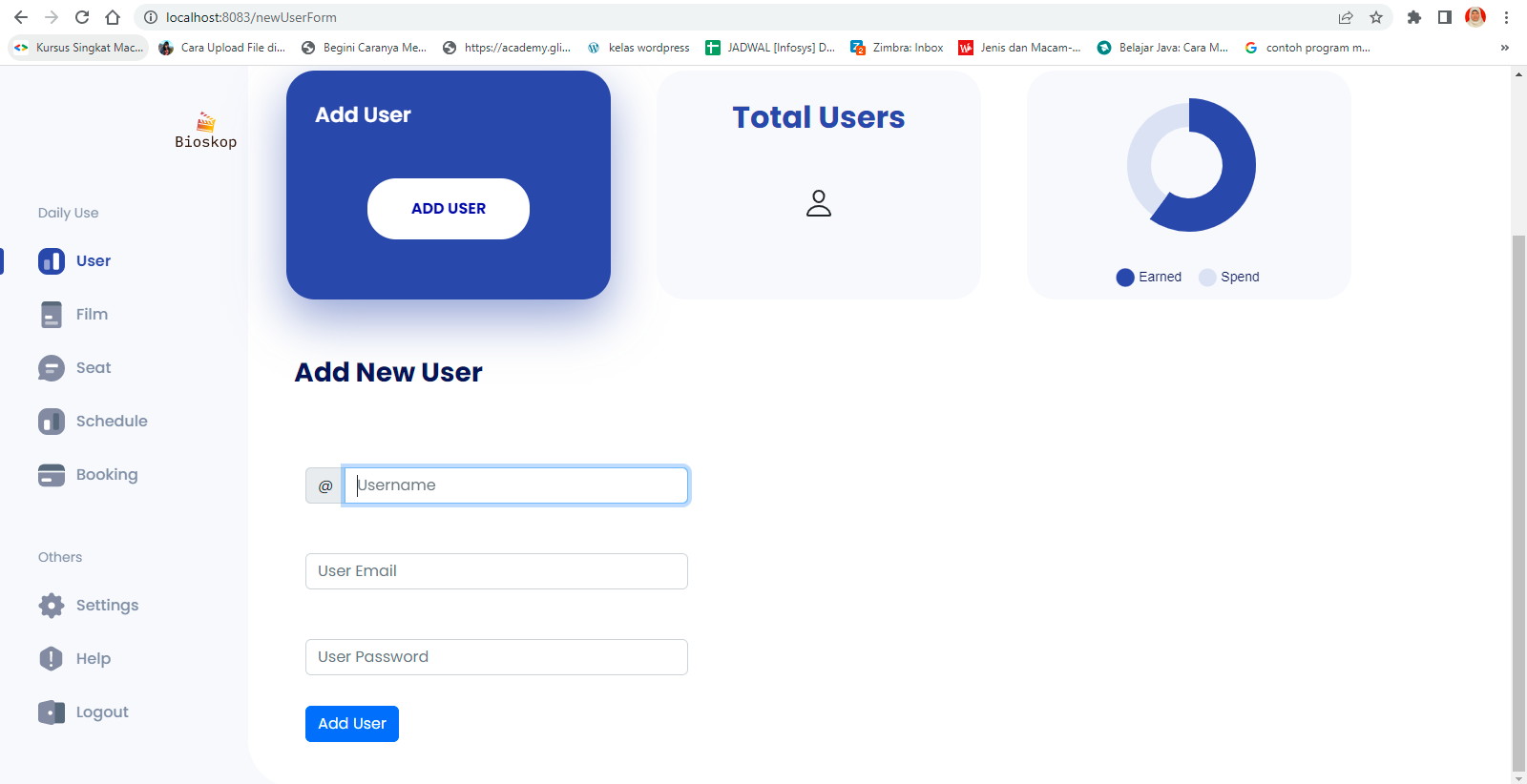
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| userid | Long | M |  | User Id yang ingin di delete. | 5 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| deleted | Boolean | M |  | Status perintah Delete. | true |

* 1. POST
* Tampilan add Users digunakan untuk membuat data user baru yang berdasarkan username, user email dan user password.





* End Point

:

<http://localhost:8083/swagger-ui/index.html#/user-controller/createUser>

* End Point Front End:

<http://localhost:8083/newUserForm>

* Request URL

http://localhost:8083/swagger-ui/index.html#/user-controller/createUser

* Sample Request

{

"username": "Luffy",

"emailId": ["luf](mailto:luffybajaklaut@onepiece.com)[fybajaklaut@onepiece.com",](mailto:fybajaklaut@onepiece.com) "password": "1231231",

"userId": 101,

"id": 0

}

* Sample Response Request

}

"data": [

{

"ID ": 61,

"Username ": "Diego",

"Email ": ["string"](mailto:luffybajaklaut@onepiece.com),

"Password ": “string"

}

],

"message": "Successfully Created User!", "status": 201

{

Success Response

{

"data": "User Already Exist!",

"message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",

"status": 400

}

Failed Response

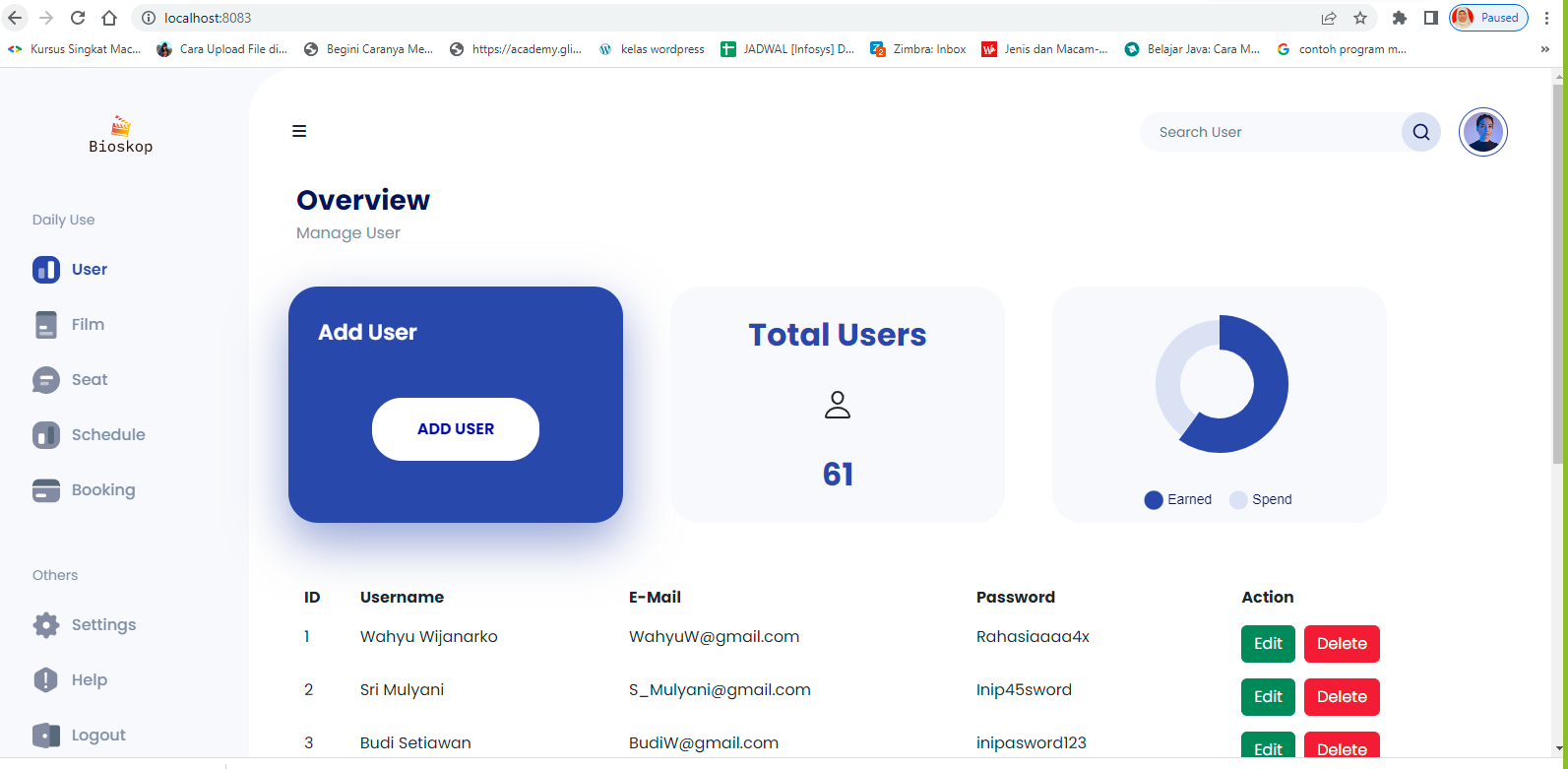
* Request Structure

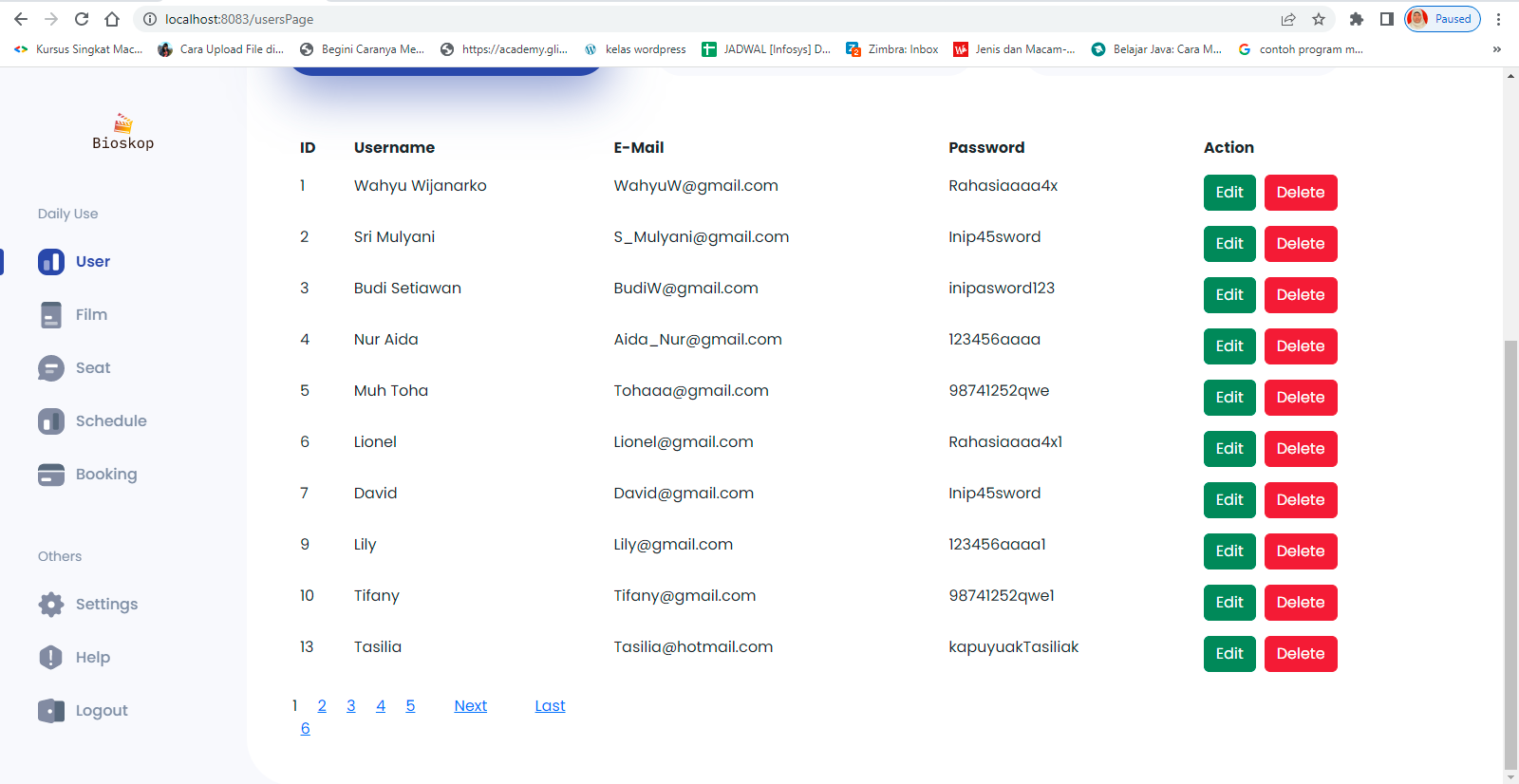
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name konsumen baru  yang ingin diinsert. | Luffy |
| emailId | String | M |  | Email konsumen baru yang ingin diinsert. | luffybajakla ut@onepie  ce.com |
| password | String | M |  | Password konsumen baru  yang ingin diinsert. | 1231231 |
| userId | Long | M |  | User Id konsumen baru yang  ingin diinsert. | 101 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name konsumen yang  berhasil diinsert. | Luffy |
| emailId | String | M |  | Email konsumen yang berhasil diinsert. | luffybajakla ut@onepie  ce.com |
| password | String | M |  | Password konsumen yang  berhasil diinsert. | 1231231 |
| userId | Long | M |  | User Id konsumen yang  berhasil diinsert. | 101 |

* 1. GET ALL
* Tampilam Users untuk menampilkan data users atau customer berdasarkan ID, username, email, password dan tombol edit serta tombol delete.





* End Point :

<http://localhost:8083/swagger-ui/index.html#/user-controller/getAllUser>

* End Point Front End:

http://localhost:8083/usersPage

* Request URL

http://localhost:8083/swagger-ui/index.html#/user-controller/getAllUser

* Sample Request
* Sample Response Request

"data": [

{

"userId": 1,

"username": "Wahyu Wijanarko",

"emailId": ["WahyuW@gmail.com"](mailto:WahyuW@gmail.com), "password": "Rahasiaaaa4x"

},

......

{

Success Response

{

"userId": 61,

"username": "Diego",

"emailId": ["string"](mailto:luffybajaklaut@onepiece.com),

"password": "string"

}

],

"message": "Successfully Get All User!", "status": 200

}

|  |
| --- |
| Failed Response |
| { |
| "data": [], |
| "message": "Tabel has no value!", |
| "status": 404 |
| } |

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| username | String | M |  | User name yang dimiliki oleh  konsumen. | Lily |
| emailId | String | M |  | Email yang digunakan oleh  konsumen saat mendaftar. | Lily@gma  il.com |
| password | String | M |  | Password yang diinput oleh  konsumen saat mendaftar. | 123456aaa  a1 |
| userId | Long | M |  | User Id yang diberikan system  untuk konsumen. | 9 |

2. Seats

* 1. GET By ID
* End Point :

http://localhost:8083/swagger-ui/index.html#/seats-controller/getseatsById

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/getseatsById

* Sample Request

{

seatId : 1

}

* Sample Response Request

}

"data": {

“seatId “1,

“seatNumber “:1,

“studioName “:: A”

“isAvailable “: 1

};

"message": "Successfully retrieved data!”!,

"status": 200

}

{

Success Response

}

"data": "Data Not Found",

"message": " Seats not Exist with id :500", "status": 404

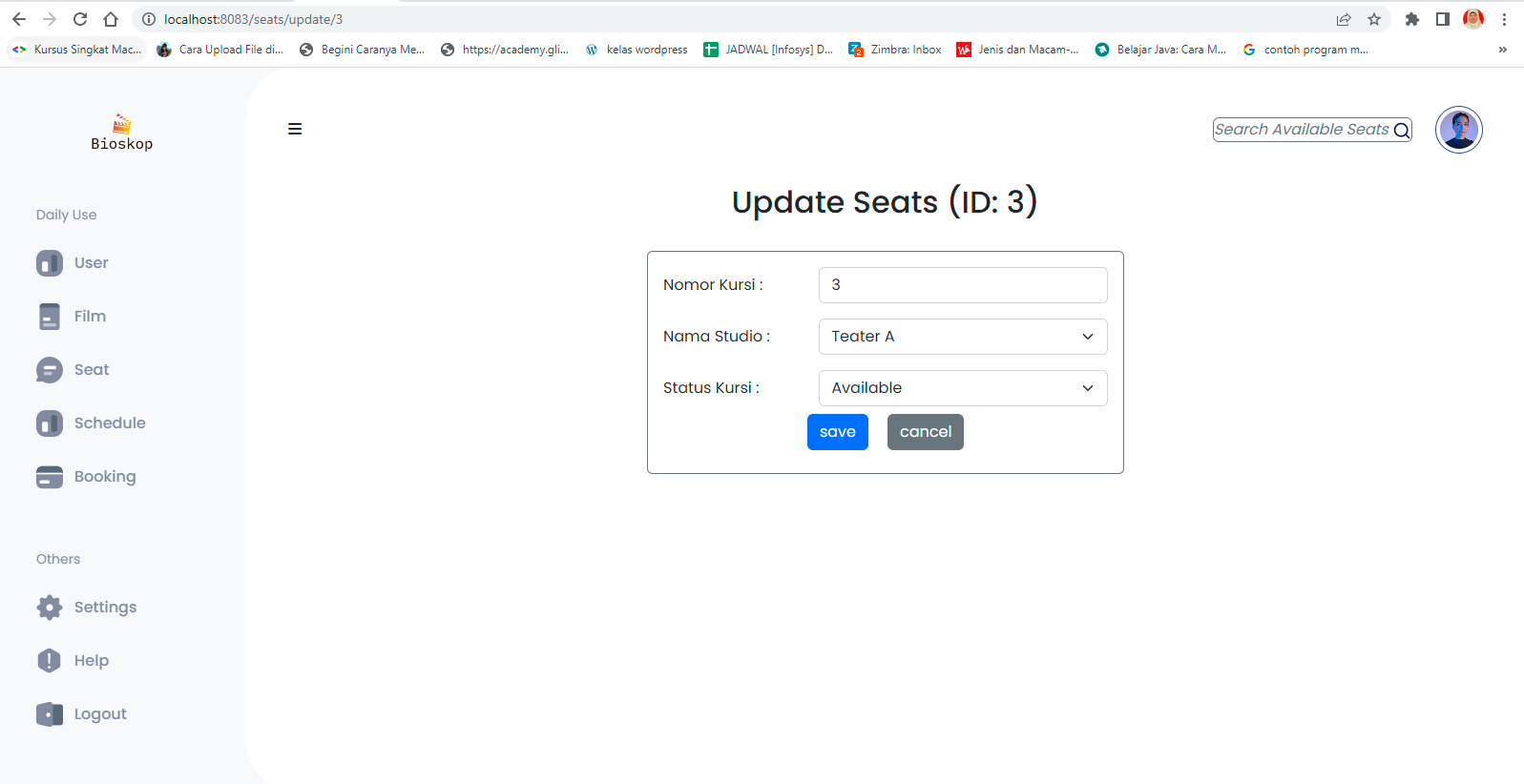
{

Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Nomor Identifikasi seat dari  system. | 1 |
| seatNumber | Long | M |  | Nomor Seat di Studio. | 1 |
| studioName | String | M |  | Nama Studio. | A |
| isAvailable | int | M |  | Status tersedia atau tidak, | 1 |

* 1. PUT
* Tampilan update seats digunakan untuk merubah Nomor Kursi, Nama Studio dan Status Kursi.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/updateSeats>

* End Point Front End:

<http://localhost:8083/seats/update/3>

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/updateSeats

* Sample Request

{

"seatId": 10,

"seatNumber": 4,

"studioName": "string", "isAvailable": 0

}

* Sample Response Request

}

"data": {

"seatId": 10,

"seatNumber": 4,

"studioName": "string", "isAvailable": 0

},

"message": "Success Update Seats", "status": 200

{

Success Response

}

"data": "data not found",

"message": " Seats not Exist with id :500", "status": 404

{

Failed Response

* + Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Seats Id kosumen yang ingin di  update. | 1 |
| seatNumber | Long | M |  | Nomor Seats konsumen yang | 2 |

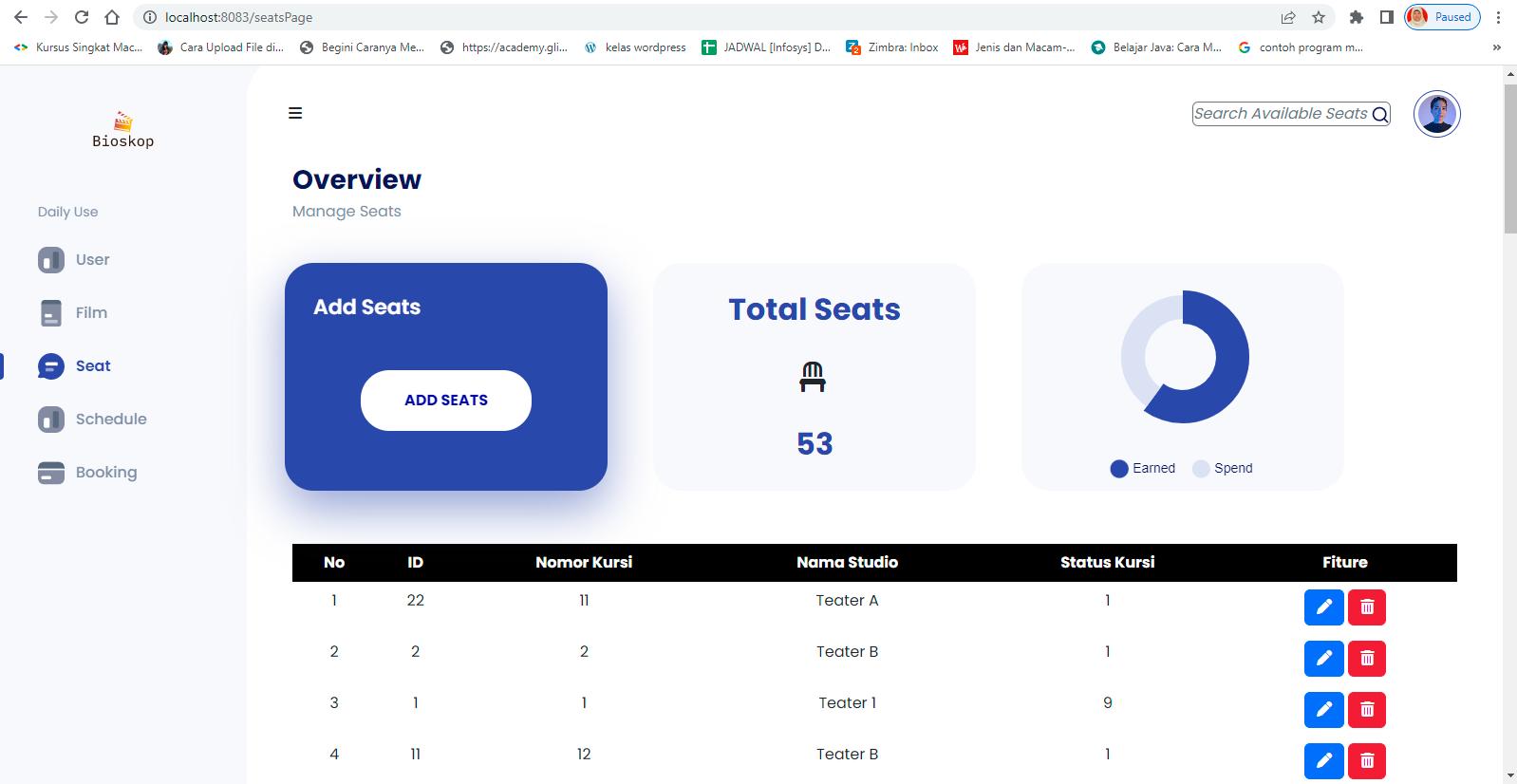
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | ingin di update. |  |
| studioName | String | M |  | Nama Studio konsumen yang  ingin di update. | C |
| isAvailable | int | M |  | Status Seats konsumen yang  ingin di update. | 0 |

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Seats Id baru yang telah  diupdate. | 1 |
| seatNumbe  r | Long | M |  | Nomor Seats yang telah  diupdate. | 2 |
| studioNam  e | String | M |  | Nama Studio yang telah  diupdate. | C |
| isAvailable | int | M |  | Status Seats konsumen yang  telah diupdate | 0 |

* 1. DELETE
* Delete

Tampilan delete untuk menghapus ID, Nomor Kursi, Nama Studio dan Status Kursi.



* End Point Front End:

http://localhost:8083/seatsPage

* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/deleteseats>

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/deleteseats

* Sample Request

{

seatId : 3

}

* Sample Response Request

"data": 3,

{

Success Response

"message": "Deleted! ", "status": 200

}

}

"data": "Data Not Found",

"message": " Seats not Exist with id :500", "status": 404

{

Failed Response

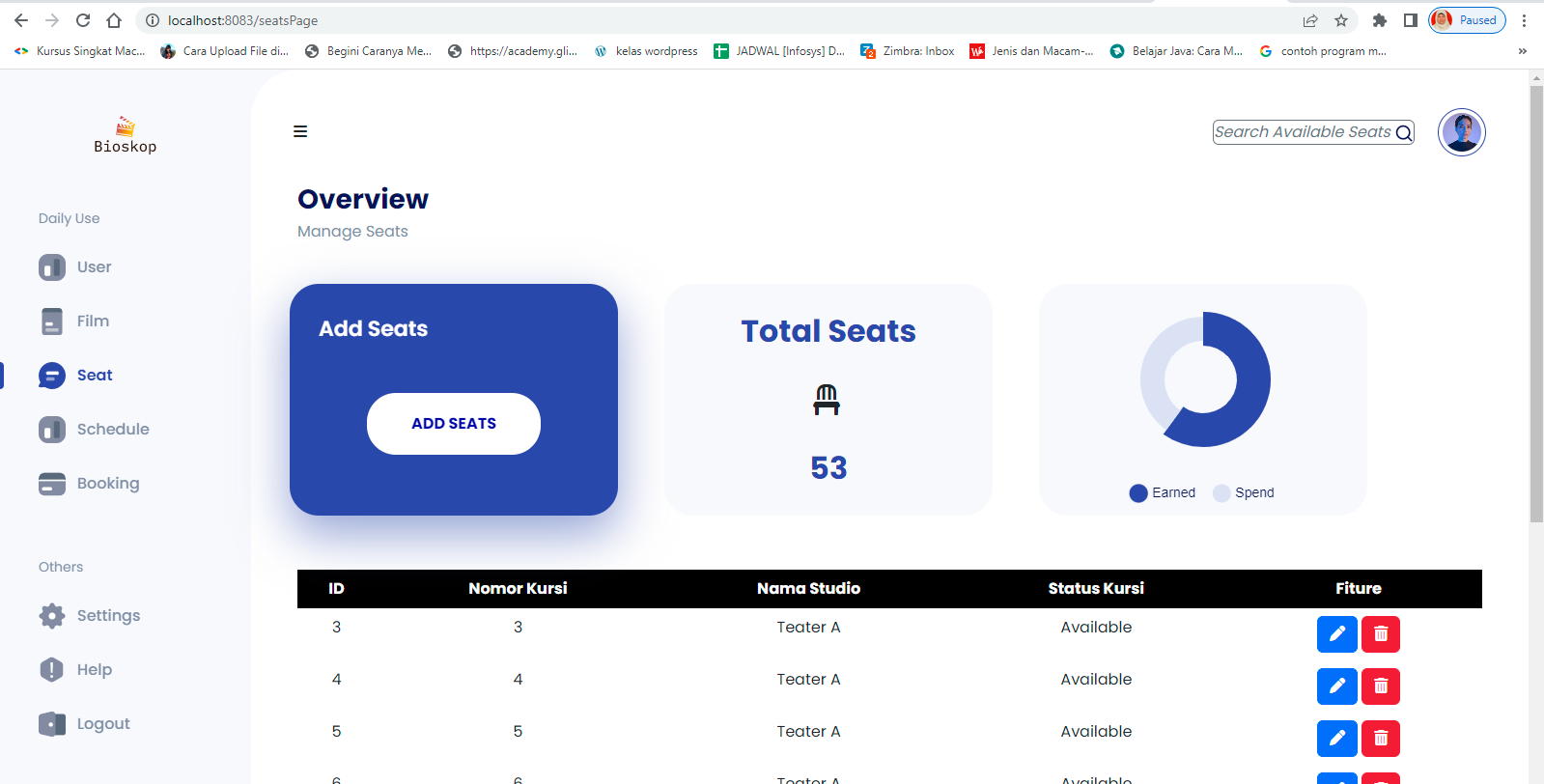
* Request Structure

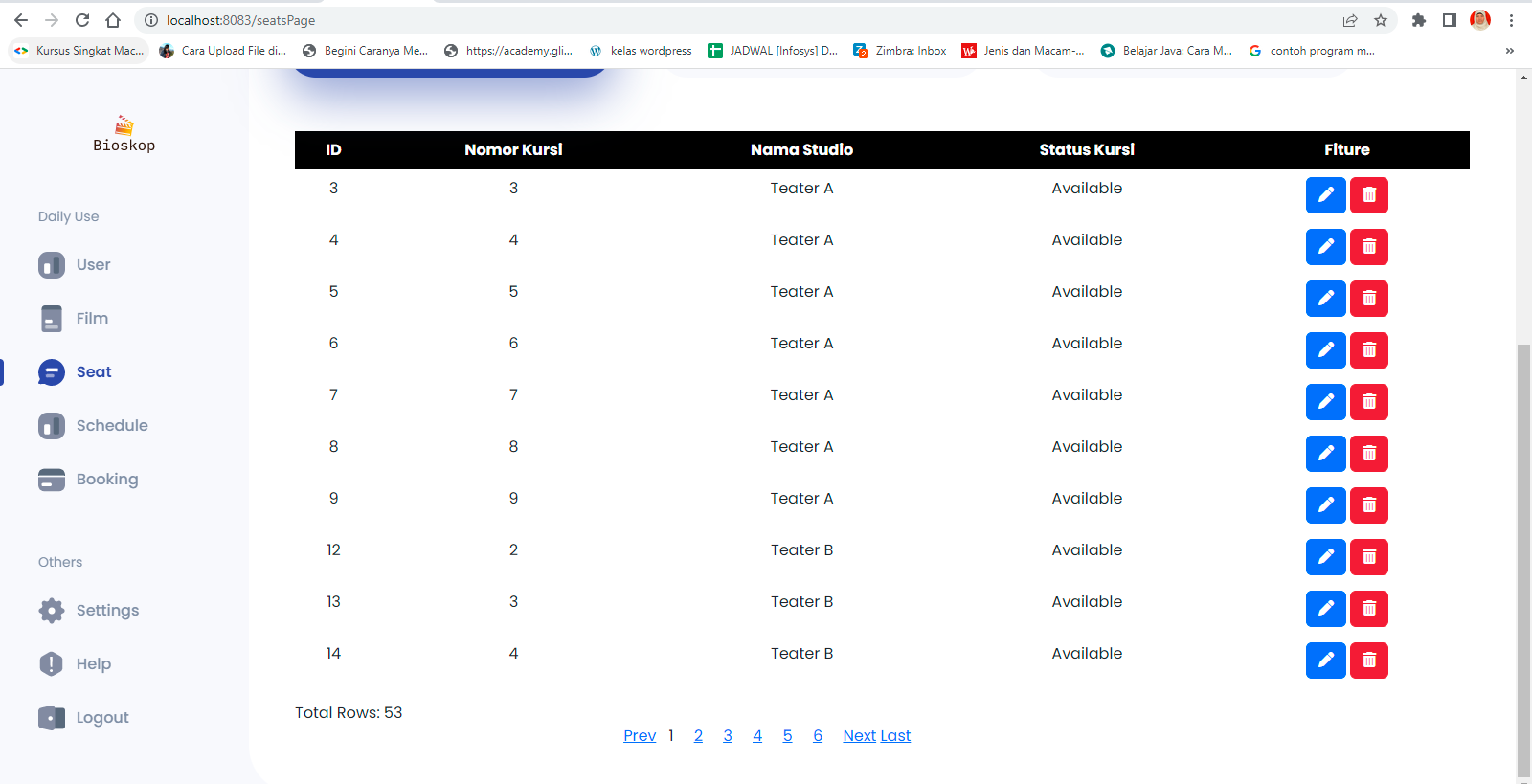
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Seat Id yang ingin di delete. | 52 |

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| deleted | Boolean | M |  | Status perintah Delete. | true |

* 1. GET ALL
* Tampilan seats untuk menampilkan nomor kursi berdasarkan ID, Nomor Kursi, Nama Studio dan Status Kursi, sedang tayang (1) atau tidak tayang (0) serta tombol add seats, update dan delete.





* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/getSeats>

* End Point Front End:

http://localhost:8083/seatsPage

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/getSeats

* Sample Request
* Sample Response Request

Success Response

{

"data": [

{

"seatId": 1,

"seatNumber": 1,

"studioName": "A", "isAvailable": 1

},

......

{

"seatId": 50,

"seatNumber": 10, "studioName": "E", "isAvailable": 1

}

],

"message": "Succesfully Read All Data Seats !", "status": 200

}

}

"data": "Table has no value",

"message": "table seats have not value", "status": 404

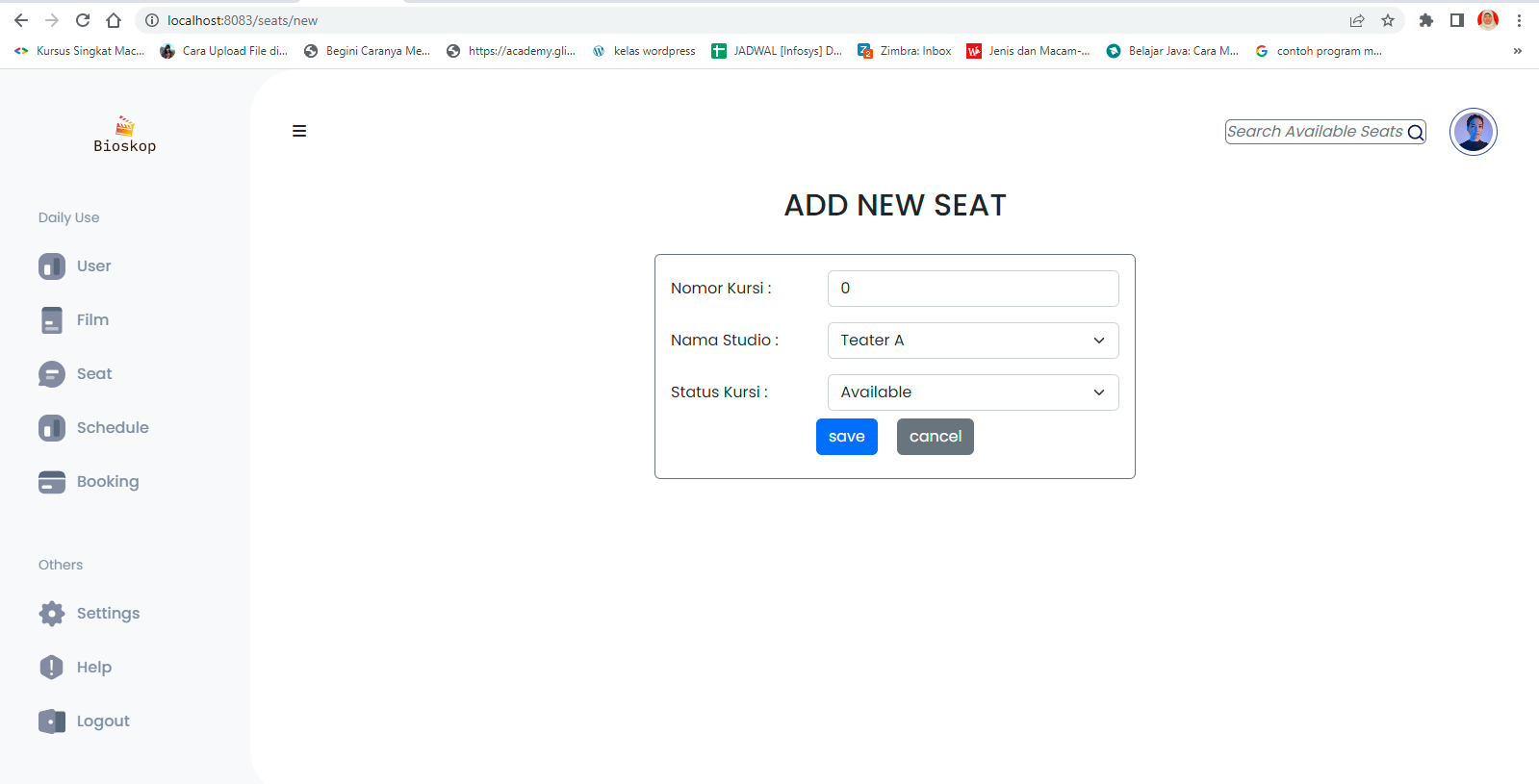
{

Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Exampl  e |
| seatId | Long | M |  | Nomor Identifikasi seat dari  system. | 2 |
| seatNumber | Long | M |  | Nomor Seat di Studio. | 2 |
| studioName | String | M |  | Nama Studio. | A |
| isAvailable | int | M |  | Status tersedia atau tidak, | 1 |

* 1. POST
* Tampilan add seats digunakan untuk menambahkan Nomor Kursi, Nama Studio dan Status Kursi.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/createseats>

* End Point Front End:

http://localhost:8083/seats/new

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/createseats

* Sample Request

{

"seatId": 51,

"seatNumber": 7,

"studioName": "A", "isAvailable": 1

}

* Sample Response Request

}

"data": {

"seatId": 2,

"seatNumber": 2,

"studioName": "A", "isAvailable": 1

},

"message": "Succesfully Add Data Seats !", "status": 200

{

Success Response

|  |
| --- |
| Failed Response |
| { |
| "data": [], |
| "message": "Tabel has no value!", |
| "status": 404 |
| } |

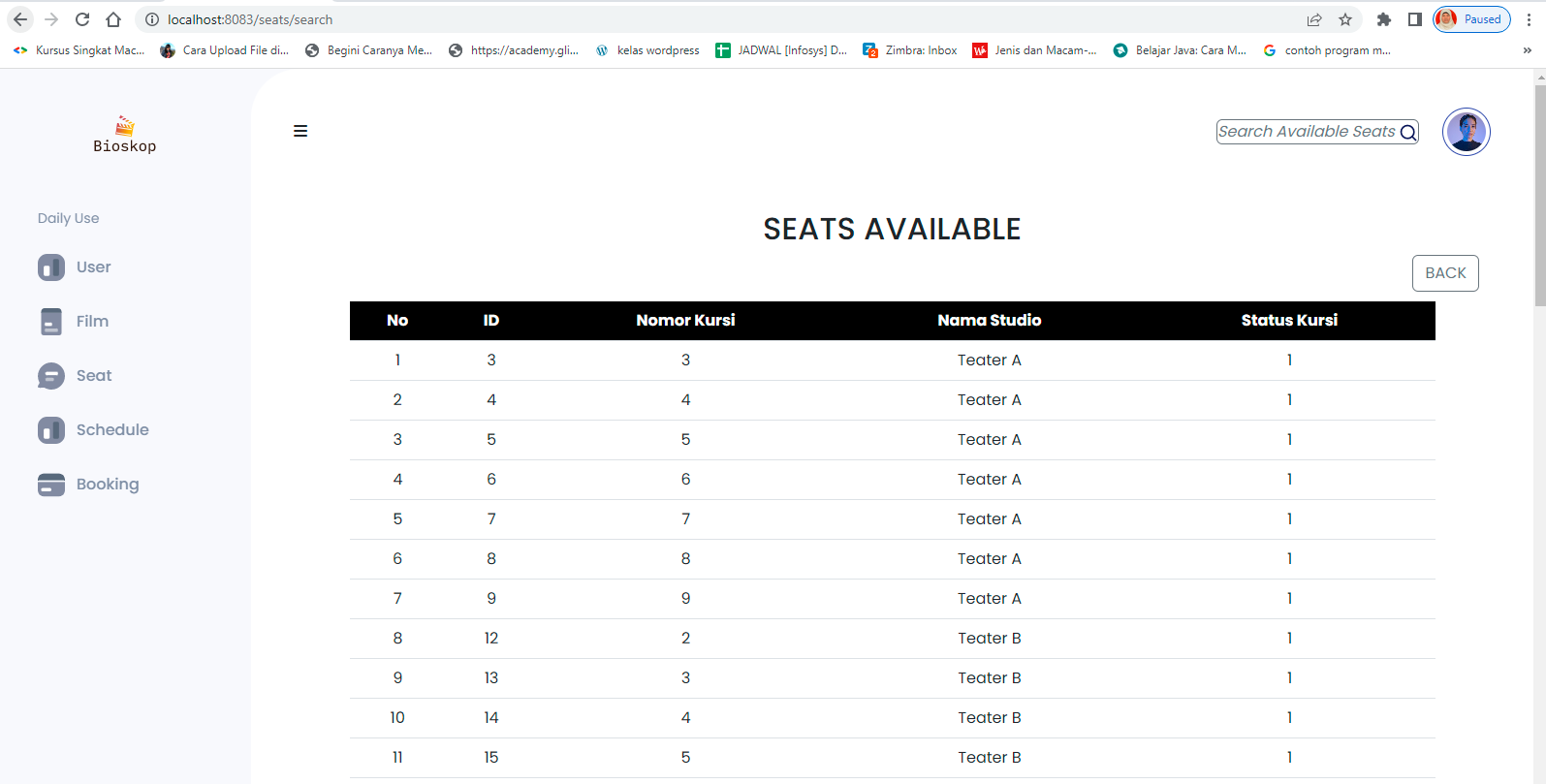
* Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Lengt  h | Description | Example |
| seatId | Long | M |  | Seats Id User name konsumen  baru yang ingin diinsert. | 52 |
| seatNumber | Long | M |  | Nomor Seats konsumen  baru yang ingin diinsert | 1 |
| studioName | String | M |  | Nama Studio konsumen  baru yang ingin diinsert | B |
| isAvailable | int | M |  | Status Seats konsumen  baru yang ingin diinsert | 1 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Seats Id User name konsumen  baru yang berhasil diinsert. | 52 |
| seatNumbe  r | Long | M |  | Nomor Seats konsumen  baru yang berhasil diinsert | 1 |
| studioNam  e | String | M |  | Nama Studio konsumen  baru yang berhasil diinsert | B |
| isAvailable | int | M |  | Status Seats konsumen  baru yang berhasil diinsert | 1 |

* 1. POST IS AVAILABLE
* Tampilan Seats Available untuk menampilkan Nomor, ID, Nomor Kursi, Nama Studio dan Status Kursi yang sedang tayang (1) atau yang sedang tidak tayang (0).



* End Point :

<http://localhost:8083/swagger-ui/index.html#/seats-controller/findSeats>

* End Point Front End :

http://localhost:8083/seats/search

* Request URL

http://localhost:8083/swagger-ui/index.html#/seats-controller/findSeats

* Sample Request

{

"seatId": 51 ,

"seatNumber": 5,

"studioName": "B", "isAvailable": 1

}

* Sample Response Request

}

"data": [

{

"seatId": 1,

"seatNumber": 1,

"studioName": "A", "isAvailable": 1,

....

{

"seatId": 50,

"seatNumber": 10, "studioName": "E", "isAvailable": 1

},

],

"message": "Successfully seats is Available ! ", "status": 200

{

Success Response

Failed Response

* Request struktur

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data  Type | Mandatory | Length | Description | Exampl  e |
| isAvailable | int | M |  | Menampilkan semua Status  Seats konsumen | 1 |

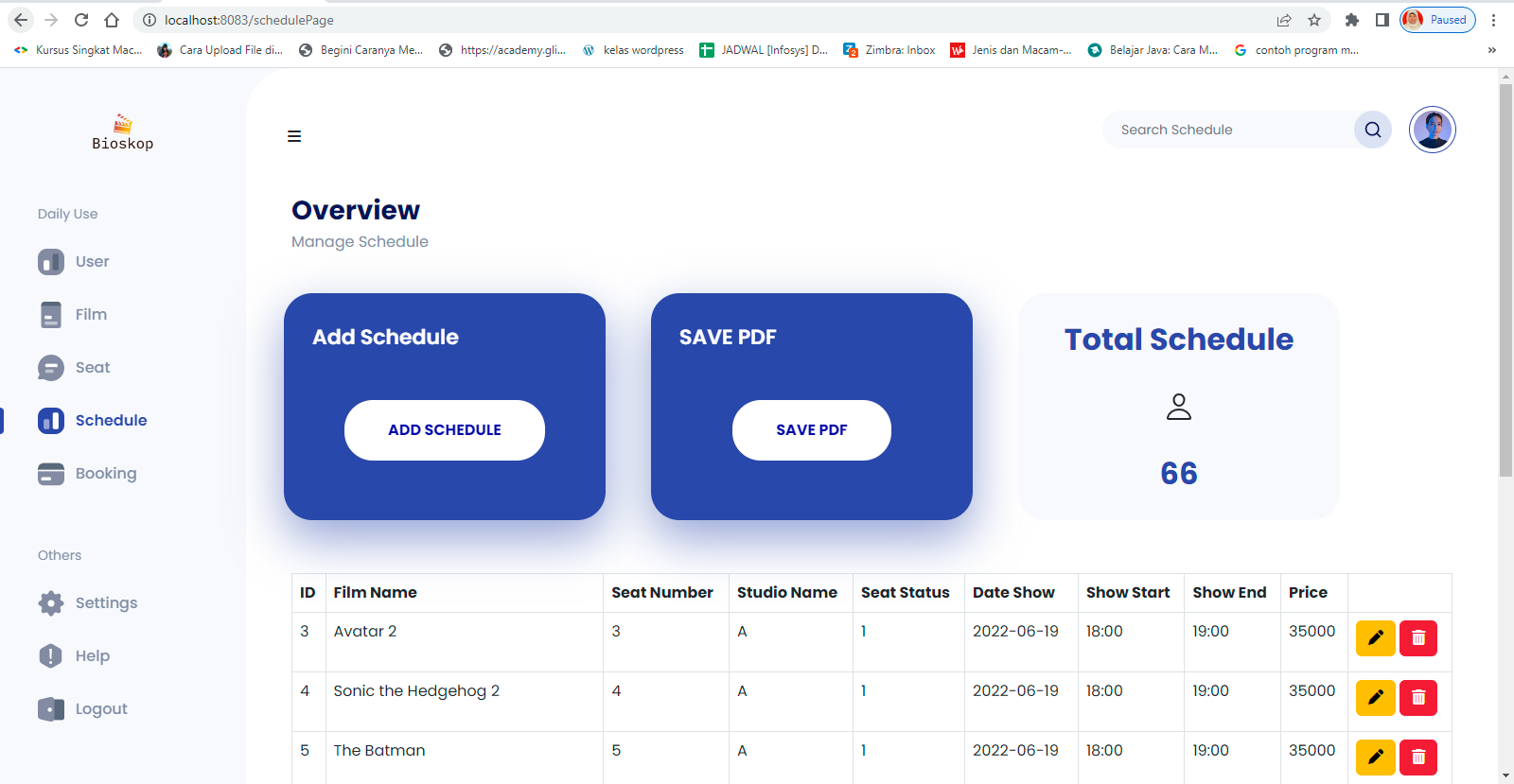
* Response Structure

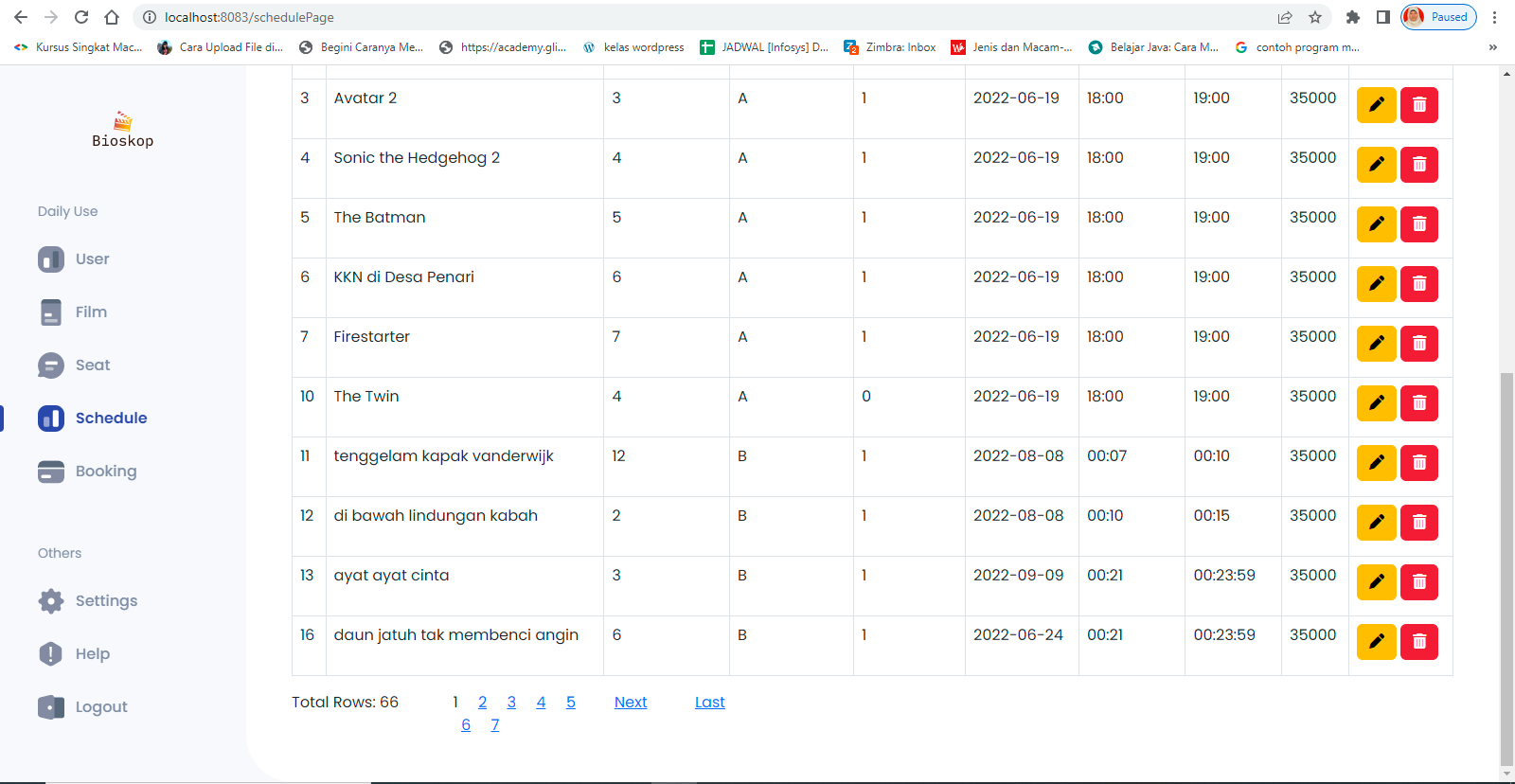
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| seatId | Long | M |  | Nomor Identifikasi seat dari  system. | 2 |
| seatNumber | Long | M |  | Nomor Seat di Studio. | 2 |
| studioName | String | M |  | Nama Studio. | A |
| isAvailable | int | M |  | Status tersedia atau tidak, | 1 |

1. SCHEDULE

3.1 GET ALL

* Tampilan Schedule untuk menampilkan jadwal film berdasarkan ID, film name, seat number, studio name , seat status, date show, show start, show end dan price serta tombol add schedule, tombol edit dan delete.





* End Point :

<http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/ScheduleList>

* End Point Front End:

http://localhost:8083/schedulePage

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/ScheduleList

* Sample Request

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/ScheduleList\_1

* Sample Response Request

}

"data": {

"scheduleId": 1,

"filmId": 1,

"seatId": 1,

"dateShow": "2022-06-19",

"showStart": "18:00:00",

"showEnd": "19:00:00",

"price": 35000

},

"message": "Successfully getAll data!", "status": 200

{

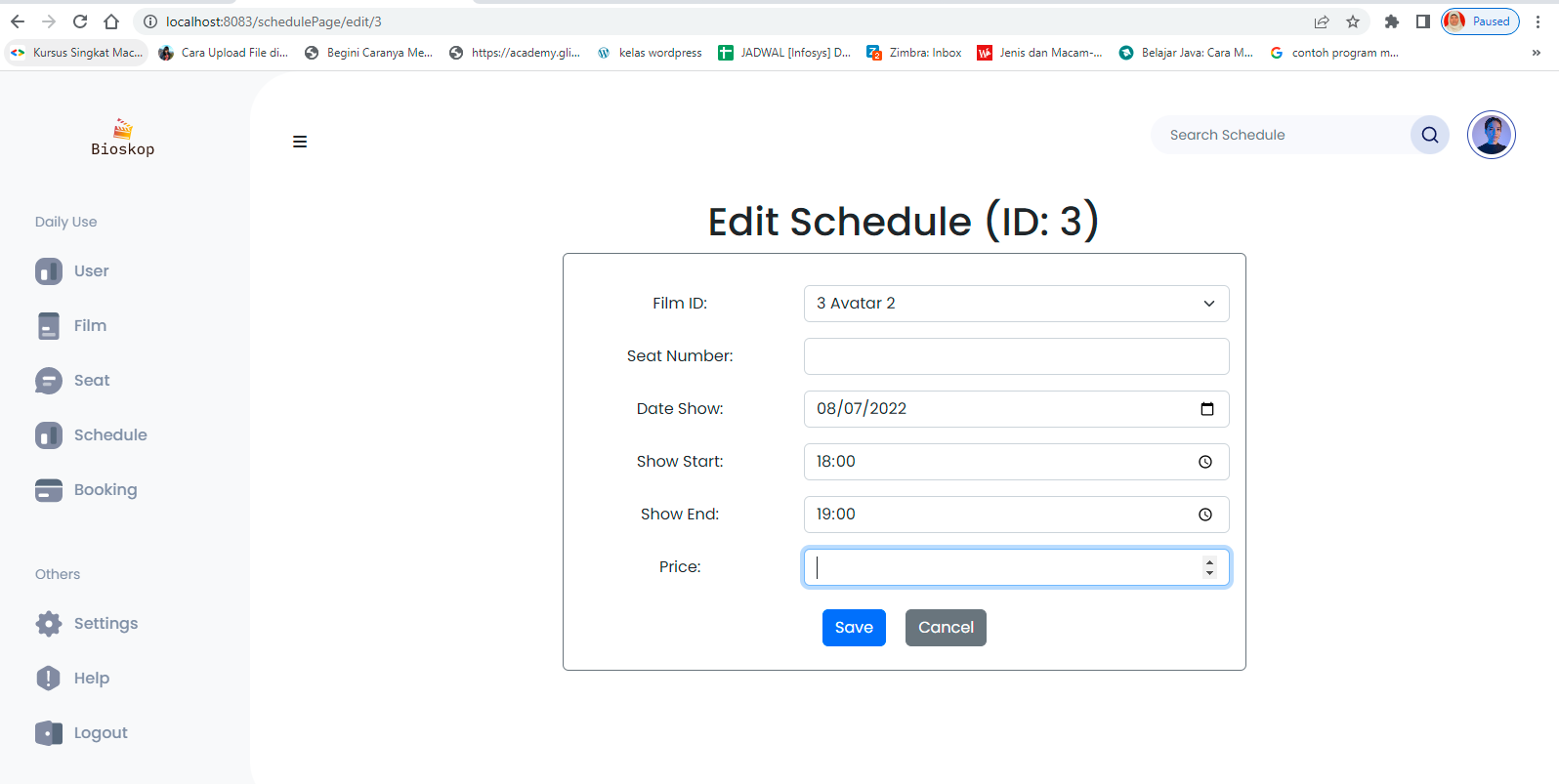
Success Response

|  |  |  |
| --- | --- | --- |
| Failed | | Response |
| { |  | |
|  | "data" "Tabel has no Value" | |
|  | "message" "Schedule not exist" | |
|  | "status" 404 | |
| } |  | |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Integer | M |  | Nomor Identifikasi  schedule dari system. | 1 |
| filmId | Integer | M |  | Nomor Identifikasi film  dari system. | 1 |
| seatId | Integer | M |  | Nomor Identifikasi seat  dari system. | 1 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-19 |
| showStart | Time | M |  | Tanggal dan waktu film  tayang. | 3:00:00 |
| showEnd | Time | M |  | Tanggal dan waktu film  selesai. | 15:00:00 |
| price | Integer | M |  | Harga tiket. | 35000 |

* 1. PUT SCHEDULE
* Tampilan Edit Schedule untuk merubah data berdasarkan film id, seat number, date show, show start, show end dan price.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/updateScheduleDTO>

* End Point Front End :

http://localhost:8083/schedulePage/edit/3

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/updateScheduleDTO

* Sample Request

{

"scheduleId": 9, "films": {

"filmId": 3

},

"seats": {

"seatId": 7

},

"dateShow": "2022-06-19",

"showStart": "00:00:00",

"showEnd": "00:00:00",

"price": 20000

}

* Sample Response Request

}

"data": {

"scheduleId": 9,

"filmId": 3,

"seatId": 7,

"dateShow": "2022-06-19",

"showStart": "00:00:00",

"showEnd": "00:00:00",

"price": 20000

},

"message": "Data Updated!", "status": 201

{

Success Response

}

"data": "Bad Request",

"message": "Schedule not exist with id : 900", "status": 400

{

Failed Response

* Resquest Structure

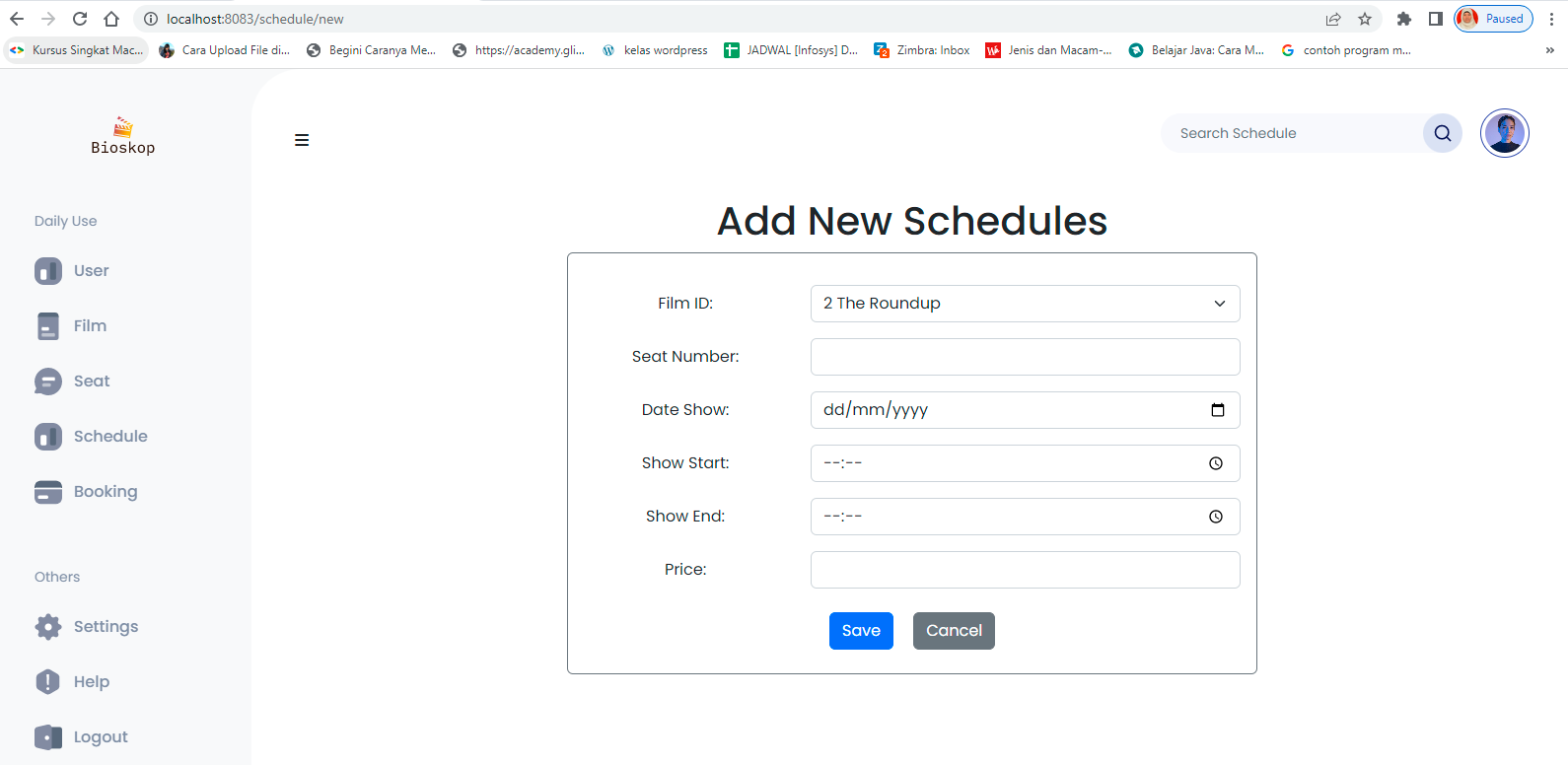
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Integer | M |  | Nomor Identifikasi  schedule dari system. | 1 |
| Films | Entity | M |  | Entity/Model dari Films |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| filmId | Integer | M |  | Nomor Identifikasi film  dari system. | 1 |
| Seats | Entitiy |  |  | Entity/Model dari seats |  |
| seatId | Integer | M |  | Nomor Identifikasi seat  dari system. | 1 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-19 |
| showStart | Time | M |  | Tanggal dan waktu film  tayang. | 3:00:00 |
| showEnd | Time | M |  | Tanggal dan waktu film  selesai. | 15:00:00 |
| price | Integer | M |  | Harga tiket. | 35000 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Integer | M |  | Nomor Identifikasi  schedule dari system. | 1 |
| filmId | Integer | M |  | Nomor Identifikasi film  dari system. | 1 |
| seatId | Integer | M |  | Nomor Identifikasi seat  dari system. | 1 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-19 |
| showStart | Time | M |  | Tanggal dan waktu film  tayang. | 3:00:00 |
| showEnd | Time | M |  | Tanggal dan waktu film  selesai. | 15:00:00 |
| price | Integer | M |  | Harga tiket. | 35000 |

* 1. POST/CREATE SCHEDULE
* Tampilan Add new schedules untuk menambahkan jadwal film dengan Film ID, Seat Number, Date Show, Show Start, Show End dan Price.



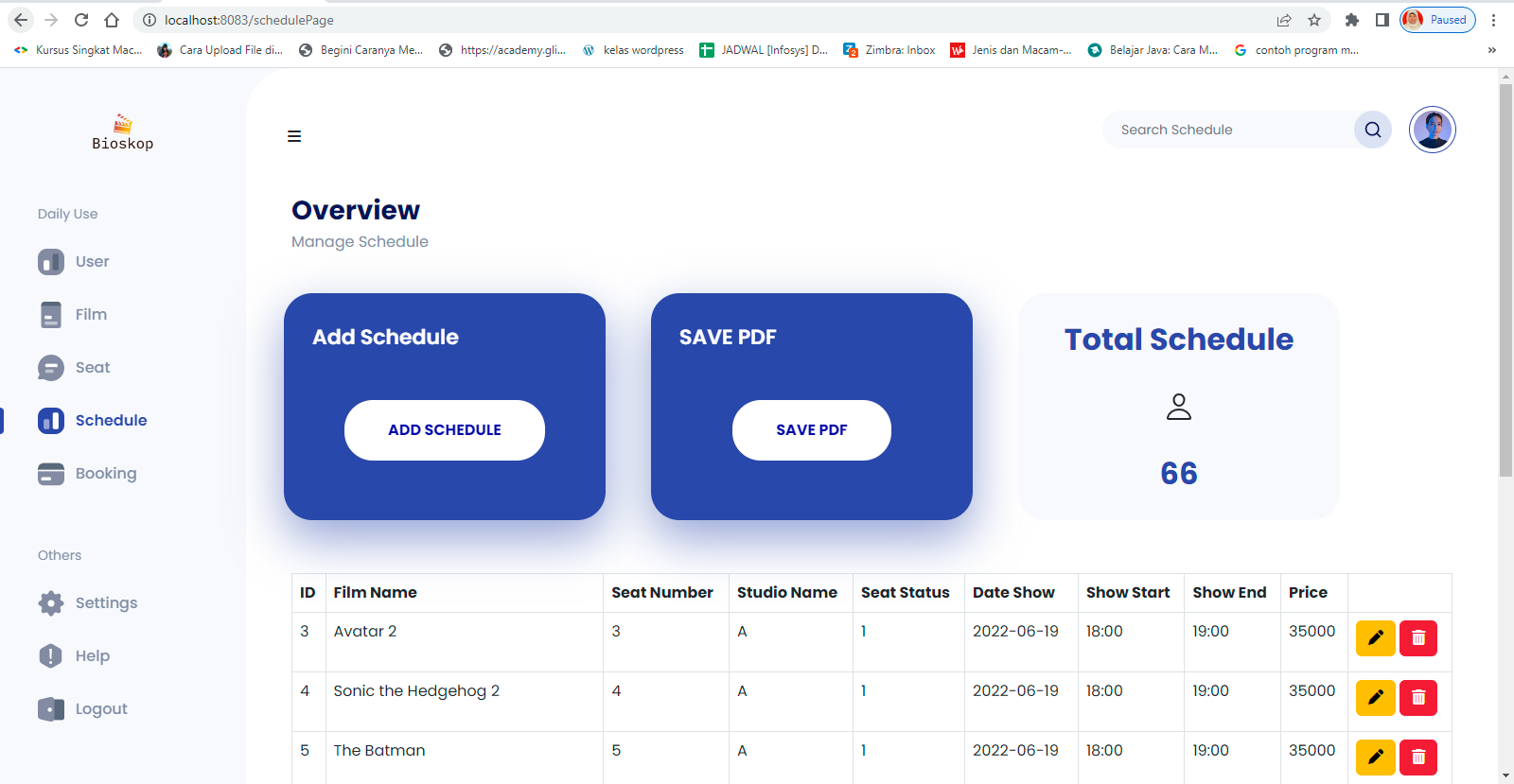
* End Point :

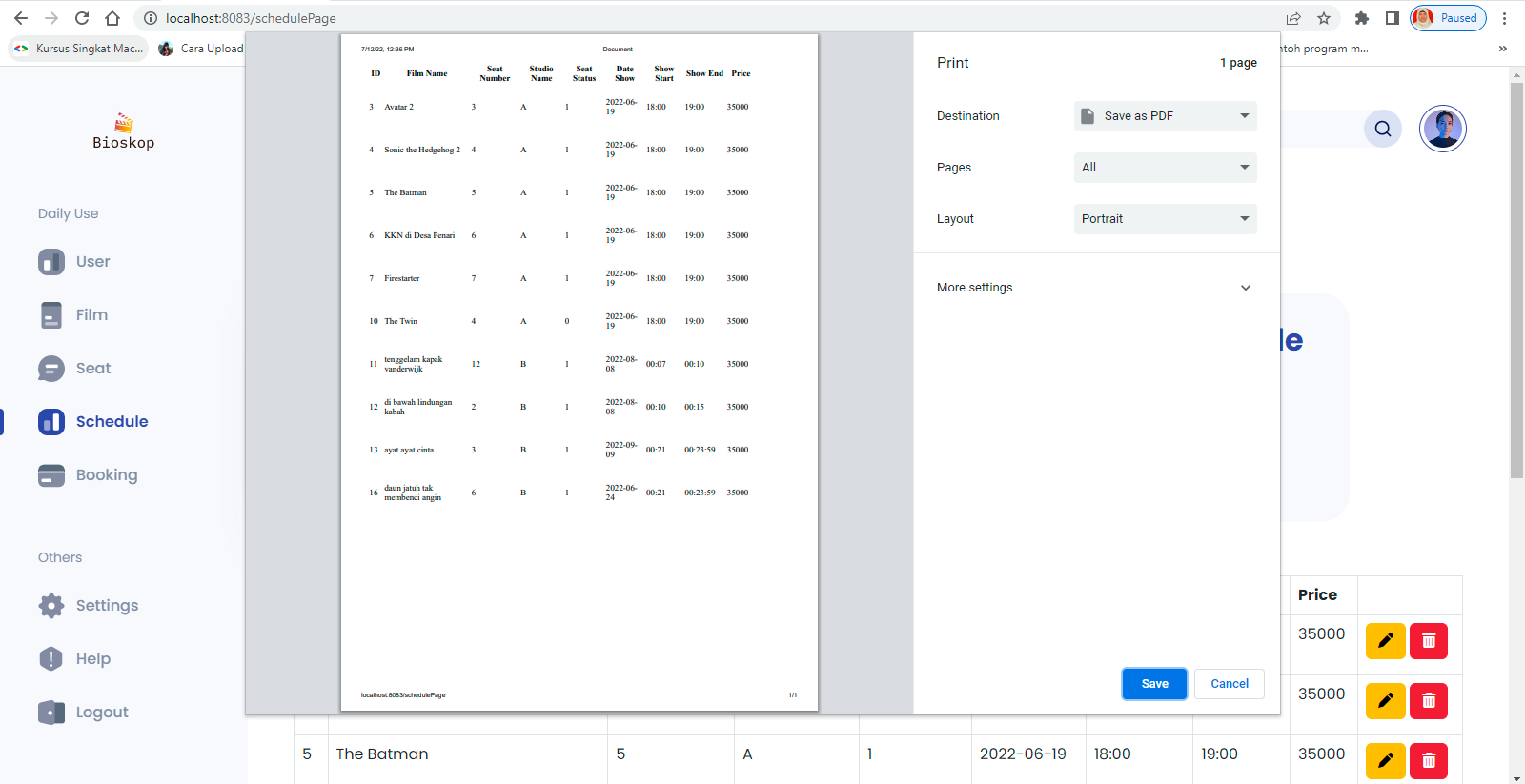
<http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/createScheduleDTO>

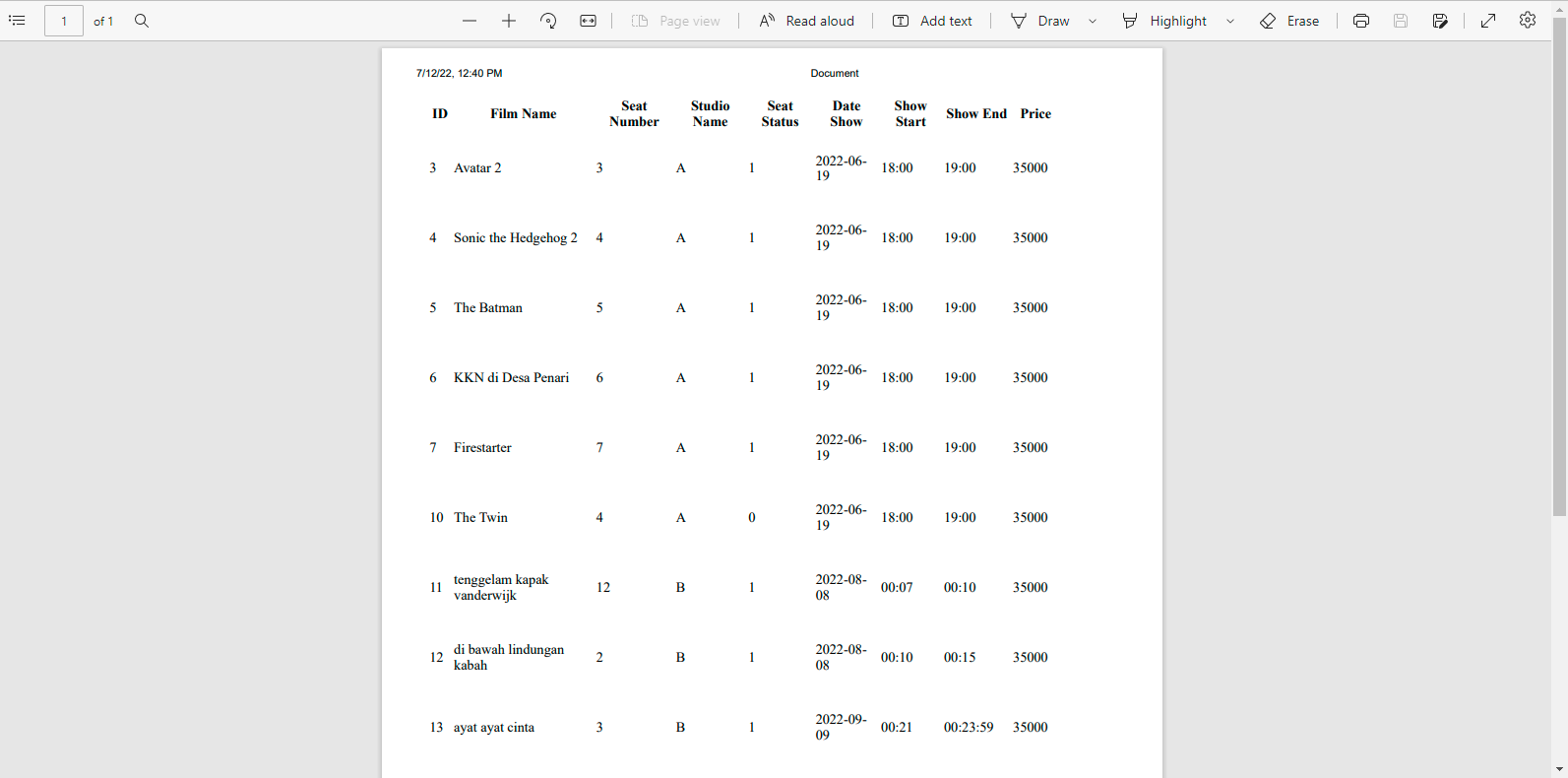
* End Point Front End:

<http://localhost:8083/schedule/new>

* Save PDF
* Tampilan save pdf untuk menyimpan data schedule yang akan disimpan.







* Endpoint Front End Save PDF:

<http://localhost:8083/schedulePage>

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/createScheduleDTO

* Sample Request

{

"films": {

"filmId": 4

},

"seats": {

"seatId": 7

},

"dateShow": "2022-06-19",

"showStart": "00:00:00",

"showEnd": "00:00:00",

"price": 20000

}

* Sample Response Request

}

"data": {

"scheduleId": 63,

"filmId": 4,

"seatId": 7,

"dateShow": "2022-06-19",

"showStart": "00:00:00",

"showEnd": "00:00:00",

"price": 20000

},

"message": "Successfully added data!", "status": 201

{

Success Response

{

"data": **null**,

"message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",

"status": 400

}

Failed Response

* Resquest Structure

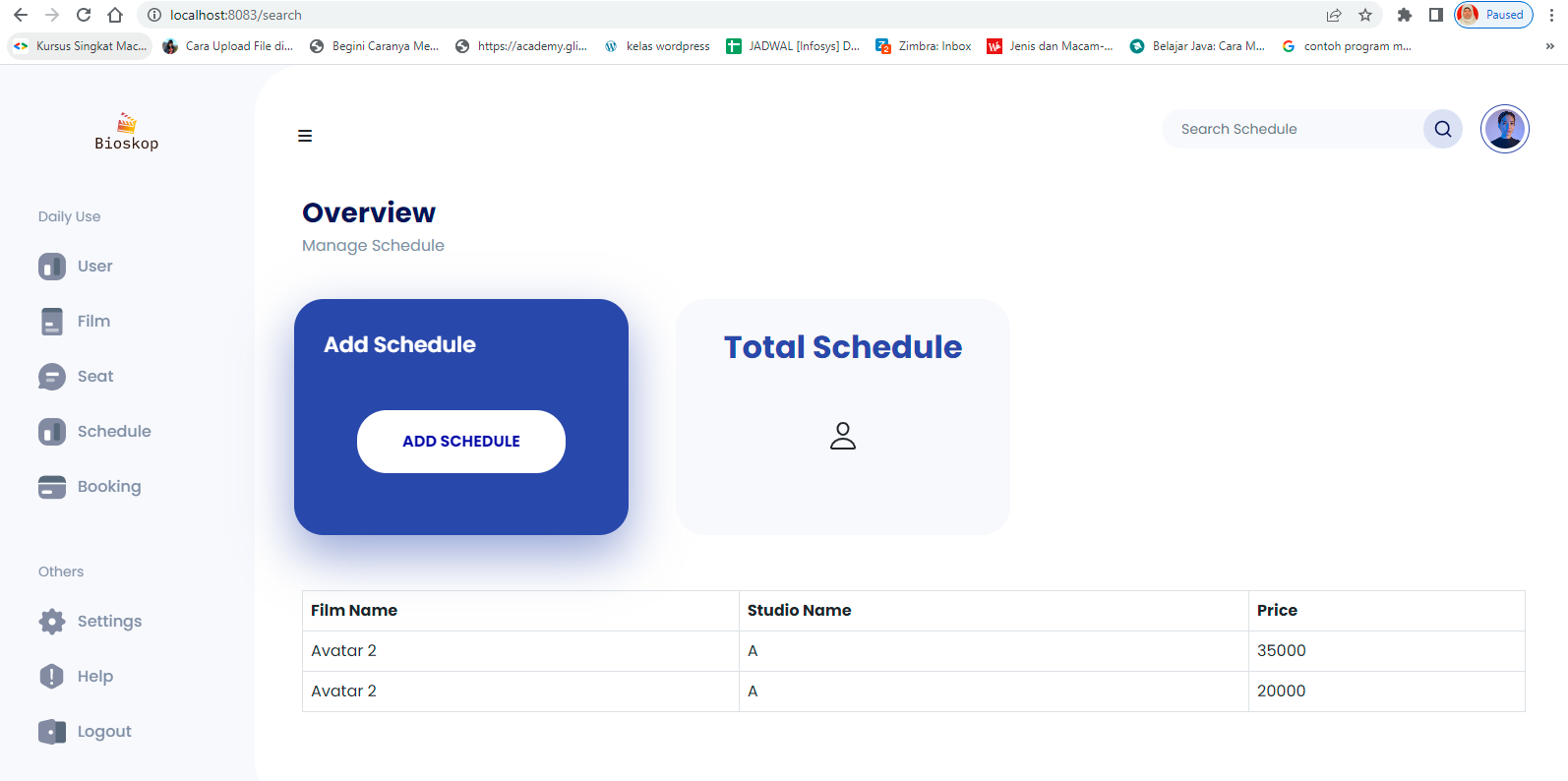
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Integer | M |  | Nomor Identifikasi  schedule dari system. | 1 |
| Films | Entity | M |  | Entity/Model dari Films |  |
| filmId | Integer | M |  | Nomor Identifikasi film  dari system. | 1 |
| Seats | Entitiy |  |  | Entity/Model dari seats |  |
| seatId | Integer | M |  | Nomor Identifikasi seat  dari system. | 1 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-19 |
| showStart | Time | M |  | Tanggal dan waktu film  tayang. | 3:00:00 |
| showEnd | Time | M |  | Tanggal dan waktu film  selesai. | 15:00:00 |
| price | Integer | M |  | Harga tiket. | 35000 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Integer | M |  | Nomor Identifikasi  schedule dari system. | 1 |
| filmId | Integer | M |  | Nomor Identifikasi film  dari system. | 1 |
| seatId | Integer | M |  | Nomor Identifikasi seat  dari system. | 1 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-19 |
| showStart | Time | M |  | Tanggal dan waktu film  tayang. | 3:00:00 |
| showEnd | Time | M |  | Tanggal dan waktu film  selesai. | 15:00:00 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| price | Integer | M |  | Harga tiket. | 35000 |

* 1. POST/FIND SCHEDULE BY FILM NAME
* POST/FIND Schedule By Film Name untuk menampilkan jadwal dari film name, studio name dan price.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/findScheduleByFilmName>

* End Point Front End :

http://localhost:8083/search

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/findScheduleByFilmName

* Sample Request

{

"name": "Badman"

}

* Sample Response Request

}

"data": [

{

"filmName": “Badman", "studioName": "D",

"price": 35000

}

],

"message": "Success", "status": 200

{

Success Response

}

"data": "Bad Request",

"message": "Schedule with film name Comic 8888 is not exist!!!!!", "status": 400

{

Failed Response

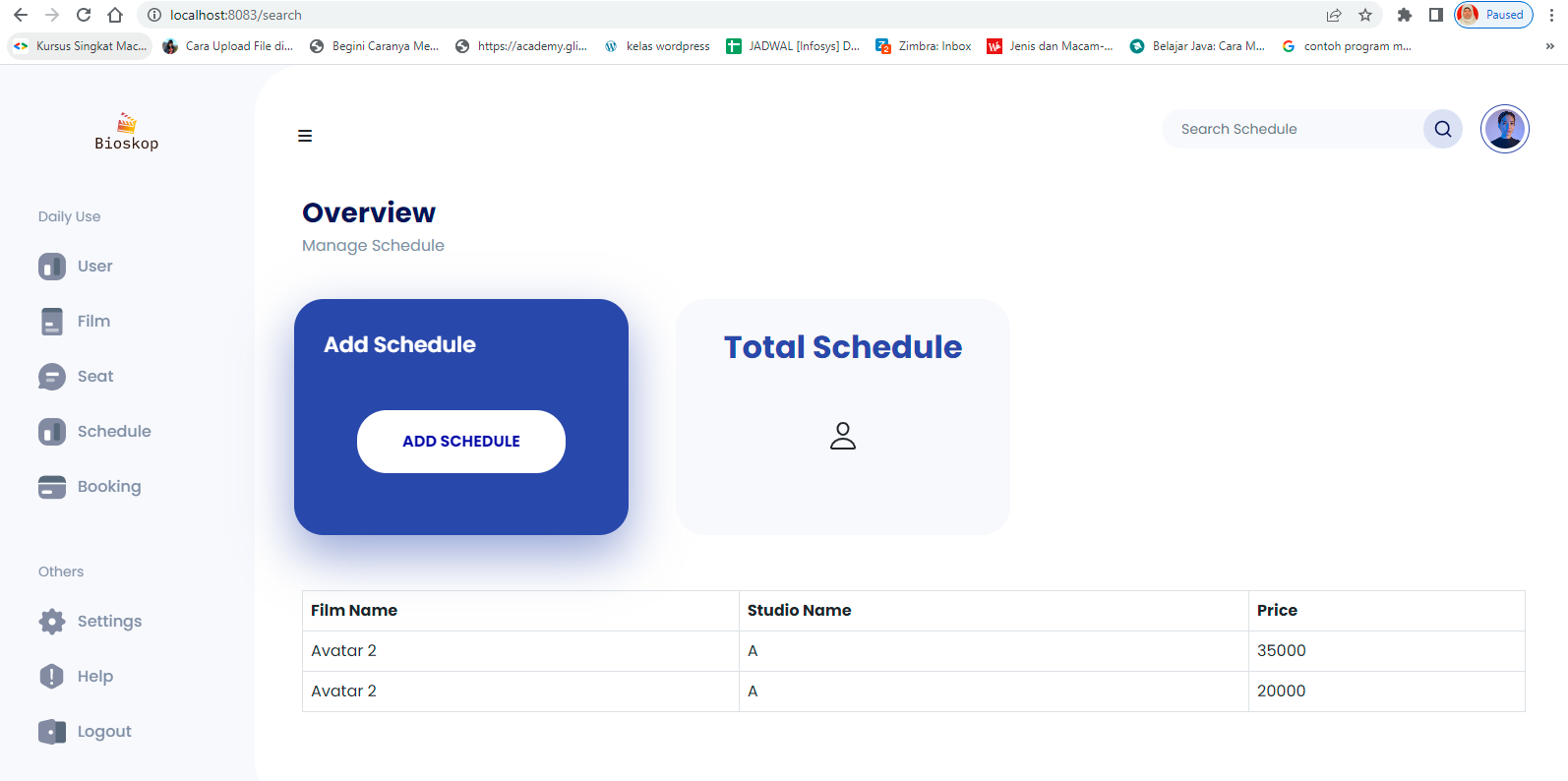
* Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| name | String | M |  | Nama film. | 1 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data  Type | Mandatory | Length | Description | Example |
| filmName | String | M |  | Nama Film | Avatar 2 |
| studioName | String | M |  | Nama Studio | A |
| price | Integer | M |  | harga Film | 25000 |

* 1. POST/ FIND SCHEDULE BY FILM NAME LIKE
* POST/FIND Schedule By Film Name Like untuk menampilkan jadwal dari film name, studio name dan price.



* End Point :

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/findScheduleByFilmName

* End Point Front End:

http://localhost:8083/search

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/findScheduleByFilmName

Sample Request

{

"name": "Avatar"

}

* + - * Sample Response Request

{

"data": [

{

"filmName": "Avatar 2", "studioName": "A",

"price": 20000

}

],

"message": "Success", "status": 200

}: 200

}

Success Response

}

"data": "Data Not Found",

"message": "Schedule with film name Comichghb jb is not exist!!!!!", "status": 404

{

Failed Response

* + - * Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| name | String | M |  | Nama film. | 1 |

* + - * Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data  Type | Mandatory | Length | Description | Example |
| filmName | String | M |  | Nama Film | Avatar 2 |
| studioName | String | M |  | Nama Studio | A |
| price | Integer | M |  | harga Film | 25000 |

* 1. GET BY ID
* End Point :

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/getscheduleById

* Request URL

<http://localhost:8083/swagger-ui/index.html#/schedule-controller->dto/getscheduleById

* Sampel Request

{

"scheduleId": 3

}

* Sampel Response

{

Success Response

"data": {

"scheduleId": 3,

"filmId": 3,

"seatId": 3,

"dateShow": "2022-06-19",

"showStart": "18:00:00",

"showEnd": "19:00:00",

"price": 35000

},

"message": "Successfully retrivied data!", "status": 200

}

}

"data": "Data Not Found!",

"message": "Schedule not exist with id : 1000", "status": 404

{

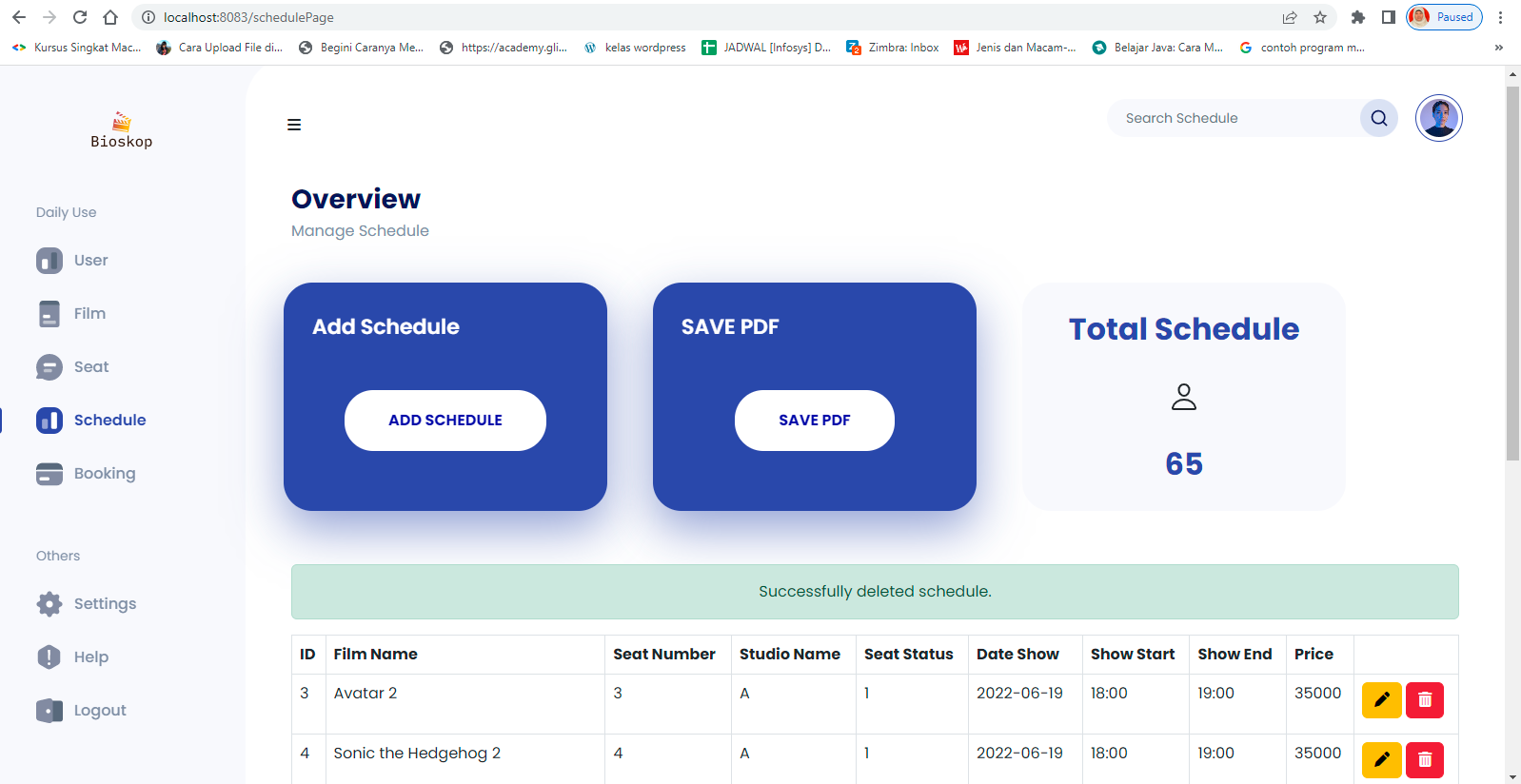
Failed Response

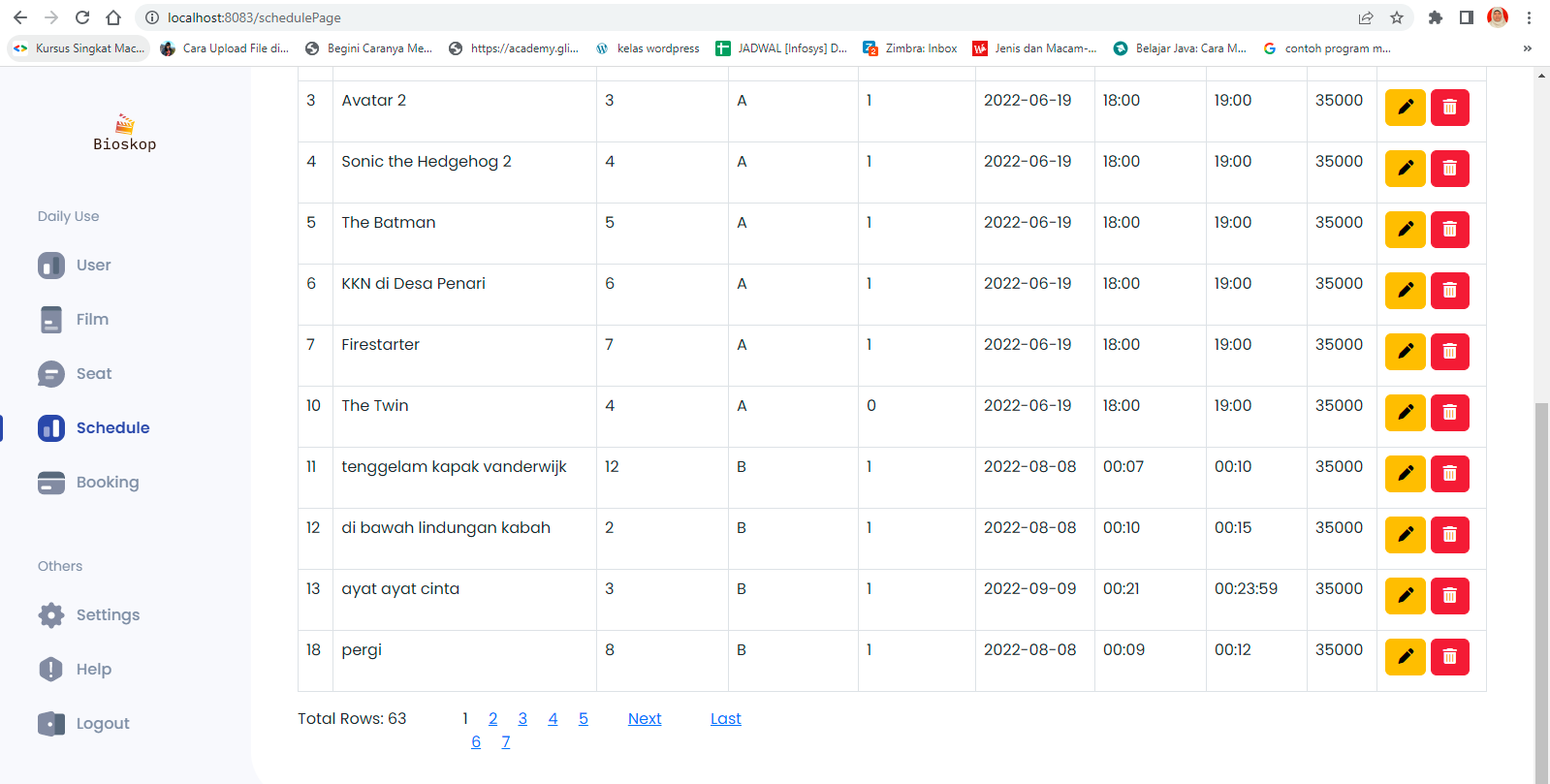
* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | int | M |  | Nomor Identifikasi  schedule dari system. | 3 |
| filmId | int | M |  | Nomor Identifikasi film  dari system. | 3 |
| name | String | M |  | Nama film. | Avatar 2 |
| isPlaying | Integer | M |  | Status sedang tayang  atau tidak. | 0 |
| dateShow | Date | M |  | Tanggal tayang. | 2022-06-  19T00:0  0:00.000  +00:00 |
| seatId | Long | M |  | Nomor Identifikasi seat  dari system. | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| seatNumb  er | Long | M |  | Nomor kursi. | 4 |
| studioNa  me | String | M |  | Nama Studio. | E |
| isAvailabl  e | int | M |  | Status kursi tersedia  atau tidak. | 0 |
| showStart | Date | M |  | Tanggal dan waktu film tayang. | 2022-06-  19T13:0  0:00.000  +00:00 |
| showEnd | Date | M |  | Tanggal dan waktu film selesai. | 2022-06-  19T15:0  0:00.000  +00:00 |
| price | Integer | M |  | Harga tiket. | 35000 |

* 1. DELETE
* Tampilan Delete untuk menghapus data berdasarkan ID, nama film, seat number, studio name, seats status, data show, show start, show end dan price.





* End Point :

<http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/deleteBooking>

* End Point Front End:

http://localhost:8083/schedulePage

* Request URL

http://localhost:8083/swagger-ui/index.html#/schedule-controller-dto/deleteBooking

* Sampel Request

{

"scheduleId": 15

}

* Sampel Response

"data": true ,

{

Success Response

},

"message": "Succes Delete Booking by ID ",

"status": 200

}

{

"data": "Data Not Found",

"message": "Schedule not exist with id :15", "status": 404

}

Failed Response

* Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| scheduleId | Long | M |  | Seat Id yang ingin di delete. | 13 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| deleted | Boolean | M |  |  | true |

4. FILM

### 4.1 GET BY ID

* End Point :

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/getfilmById

* Request URL

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/getfilmById

* Sampel Request

{

"filmId": 1

}

* Sampel Response

Success Response

{

"data": {

"filmId": 1,

"name": "Jurassic World Dominion", "isPlaying": 1

},

"message": "Succes Get", "status": 200

}

}

"data": "Data not found",

"message": "Films with id 100 is not exist", "status": 404

{

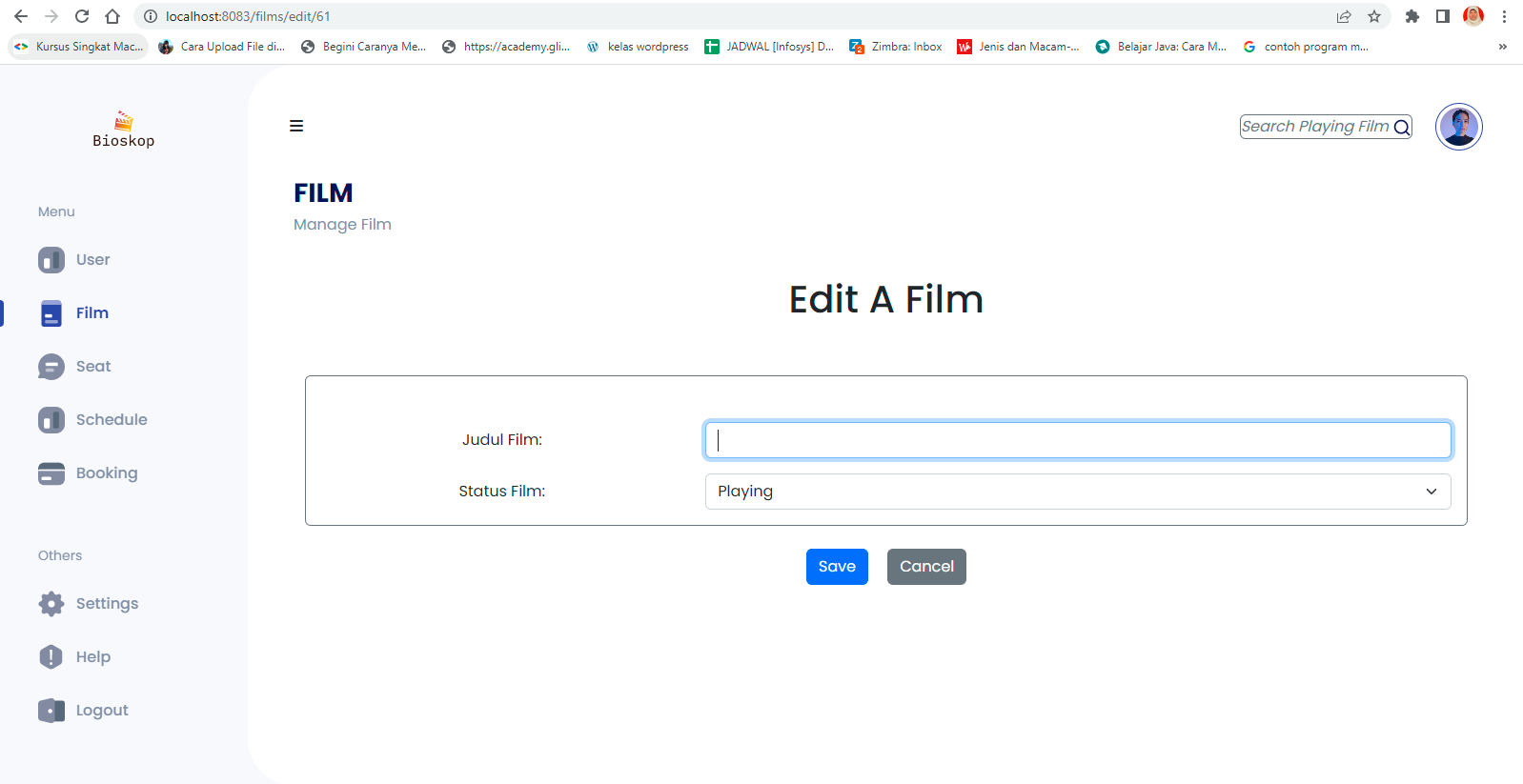
Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Nomor Identifikasi film dari  system. | 1 |
| name | String | M |  | Nama film. | Jurassic World  Dominion |
| isPlaying | Integer | M |  | Film tersedia atau tidak. | 1 |

* 1. PUT
* Tampilan edit digunakan untuk merubah data berdasarkan Judul Film dan

Status sedang tayang (1) atau status sedang tidak tayang (0).



* End Point :

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/filmsupdate>

* End Point Front End :

<http://localhost:8083/films/edit/23>

* Request URL

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/filmsupdate

* Sampel Request

{

"name":“Sun” “isplaying": 1

}

* Sampel Response

}

"data": {

"filmId": 65,

"name": "kamu", "isPlaying": 0

},

"message": "Succes Update", "status": 201

{

Success Response

}

"data": "Data not found",

"message": "Films with id 100 is can't to update", "status": 404

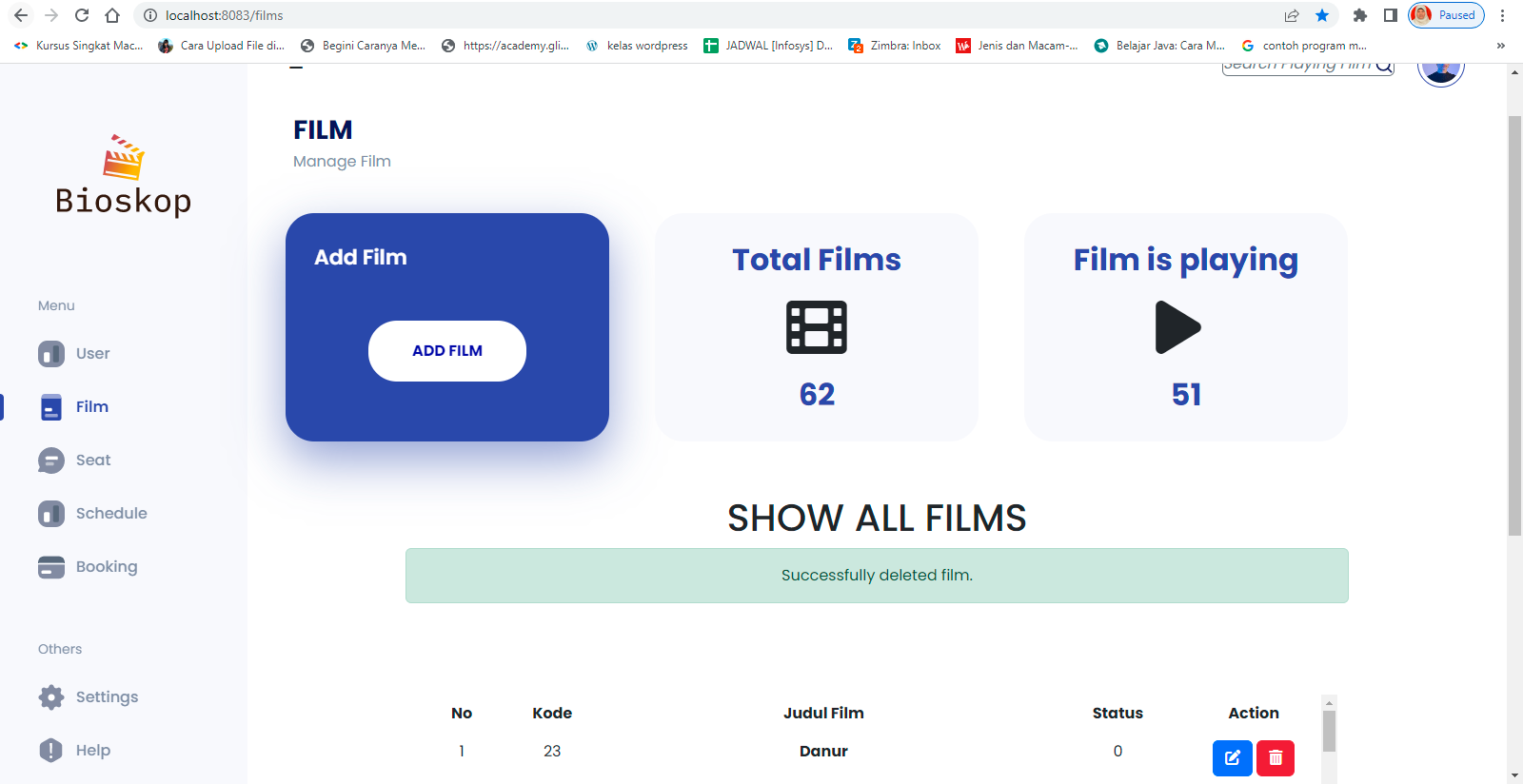
{

Failed Response

* + Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Film Id konsumen yang  Berhasi diupdate.. | 1 |
| name | String | M |  | Nama film konsumen yang  berhasil diupdate.. | kamu |
| isPlaying | Integer | M |  | Status tayang film yang  berhasil diupdate. | 0 |

* 1. DELETE
* Tampilan delete film figunakan untuk mrnghapus nomor, kode, judul Film dan status sedang tayang (1) atau status tidak tayang (0).



* End Point :

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/deleteFilm>

* End Point Front End:

http://localhost:8083/films

* Request URL

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/deleteFilm

* Sampel Request

{

"filmId": 63

}

* Sampel Response

}

"data": {

"deleted": true

},

"message": "Succes Delete", "status": 200

{

Success Response

}

"data": "Failed delete data",

"message": "Films with id 100 is failed to delete", "status": 404

{

Failed Response

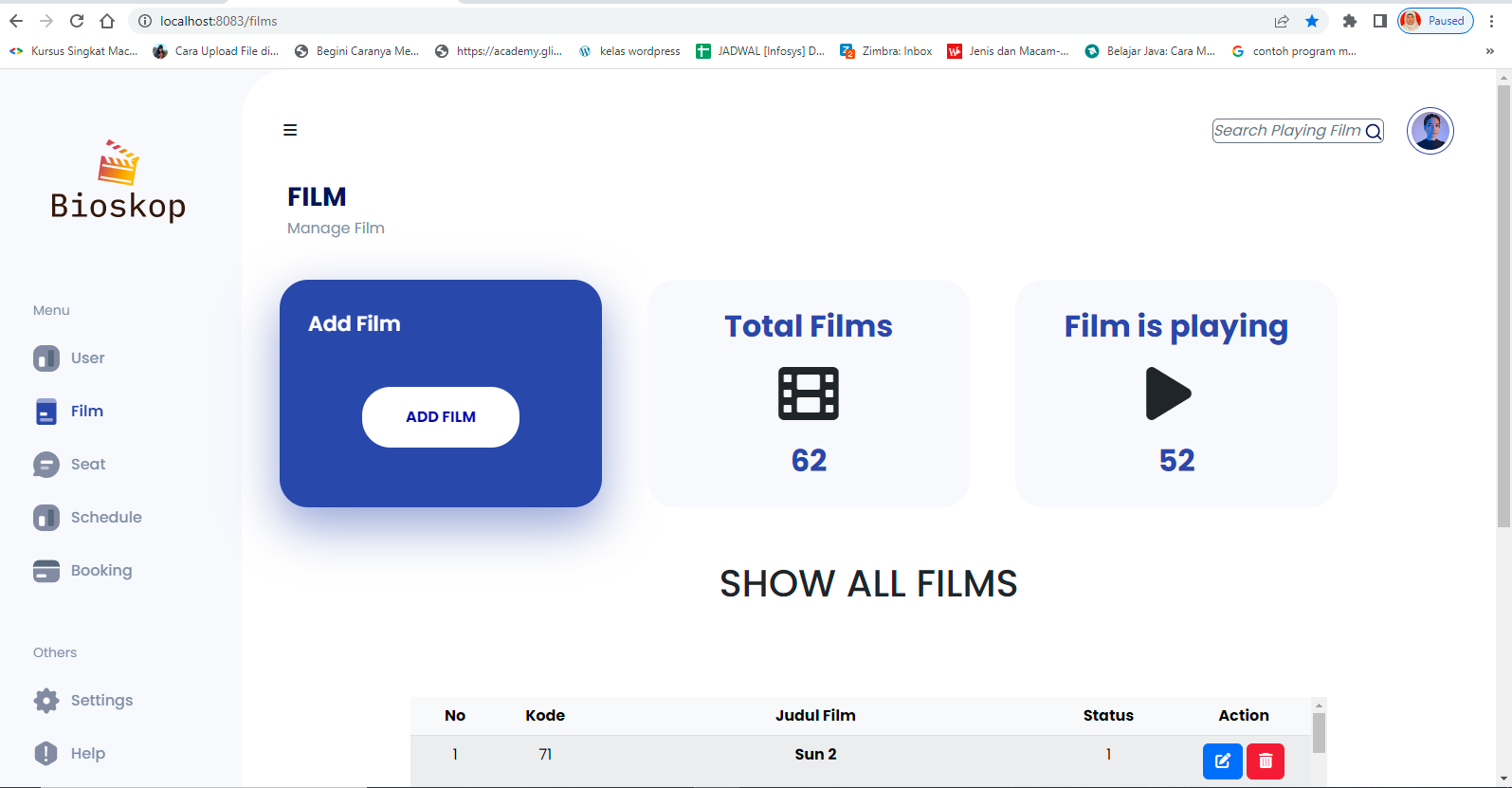
* Request Structure

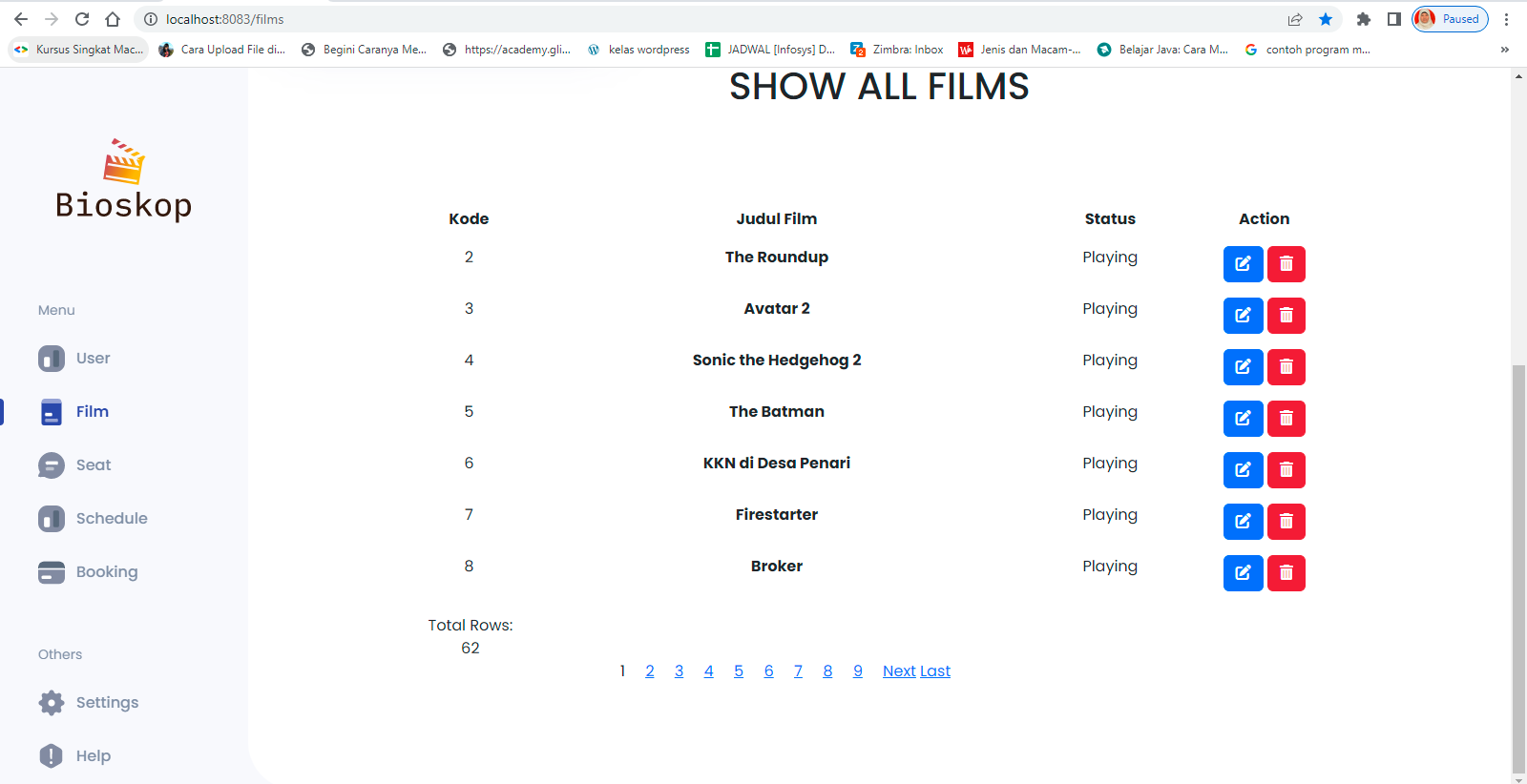
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Exampl  e |
| filmId | Long | M |  | Film Id yang ingin di delete. | 20 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| deleted | Boolean | M |  | Status perintah Delete. | true |

* 1. GET ALL
* Tampilan film digunakan untuk menampilkan nomor, kode, judul film, status sedang tayang (1) atau tidak sedang tayang (0) dan tampilan film digunakan untuk menambahkan film, mengedit film, dan menghapus film.





* End Point

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/findAllFilms>

* End Point Front End :

http://localhost:8083/films

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/findAllFilms\_1

Request URL

* Sampel Request
* Sampel Response

]

"filmId": 66,

"name": "GERAKAN UJI NYALI",

"isPlaying": 1

}

],

"message": "Succes All", "status": 200 }

{

"data": [

{

"filmId": 1,

"name": "Jurassic World Dominion", "isPlaying": 1

},

{

Success Response

}

"data": "Table has no value",

"message": "Table films has not value", "status": 404

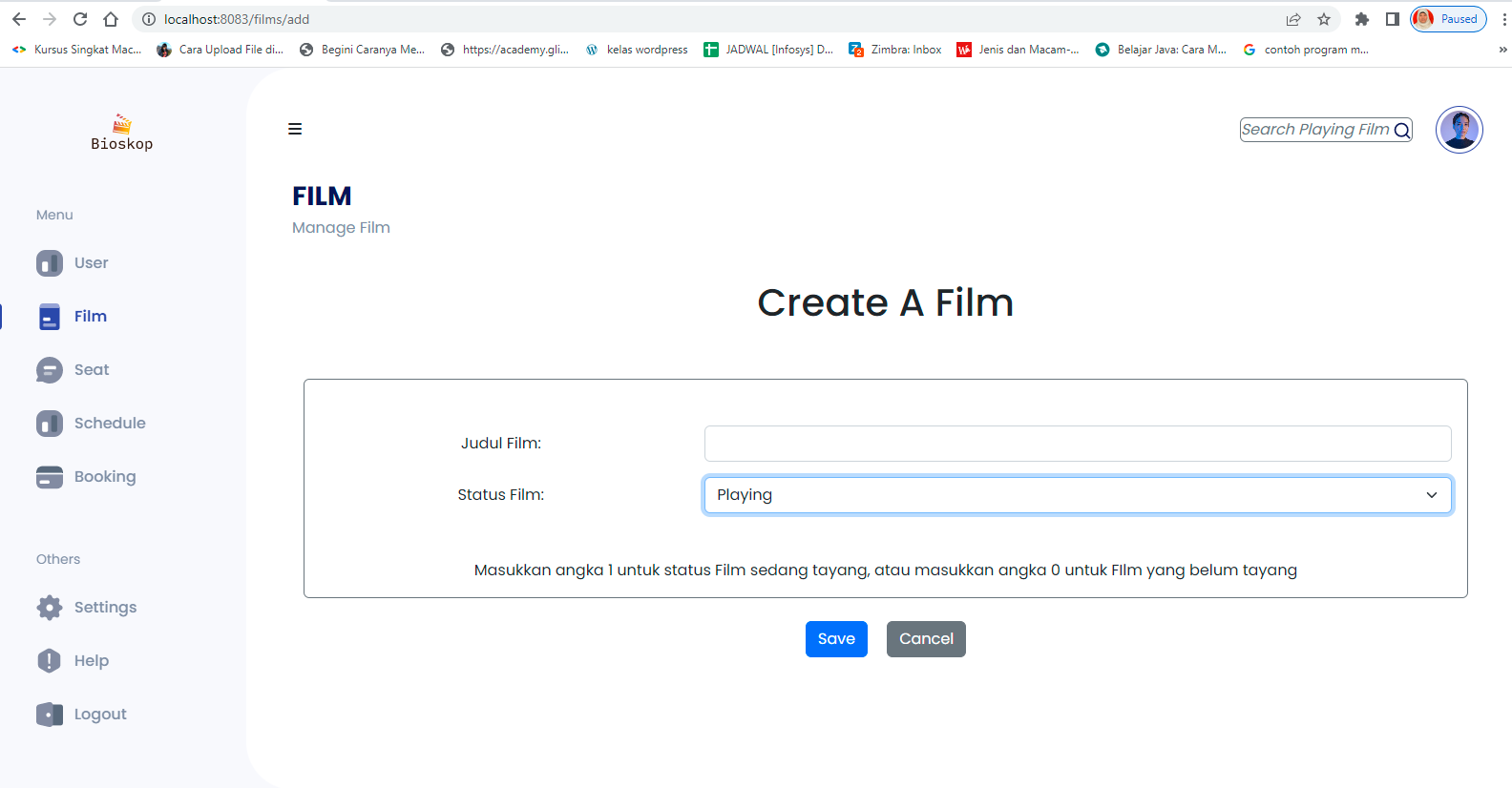
{

Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Nomor Identifikasi film dari  system. | 1 |
| name | String | M |  | Nama film. | kamu |
| isPlaying | Int | M |  | Film tersedia atau tidak | 0 |

* 1. POST
* Tampilan Add Film digunakan untuk membuat judul dan Status baru yaitu status sedang tayang (1) atau status tidak sedang tayang (0).



* End Point

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/createFilm>

* End Point Front End :

http://localhost:8083/films/add

* Request URL

http://localhost:8083/swagger-ui/index.html#/films-controller-dto/createFilm

* Sampel Request

{

|  |
| --- |
|  |

"name" "dzikir 24 hours" "isPlaying" 1

}

* Sampel Response

}

"data": {

"filmId": 69,

"name": "dzikir 24 hours", "isPlaying": 0

},

"message": "Succes Create", "status": 200

{

Success Response

{

"data": "Failed create data",

"message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",

"status": 400

}

Failed Response

* + Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Film Id konsumen yang | 62 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | ingin di insert.. |  |
| name | String | M |  | Nama film konsumen yang  ingin di insert. | dzikir 24  hours |
| isPlaying | Integer | M |  | Status tayang film tersedia tau  tidak yang ingin di insert. | 1 |

Response Body



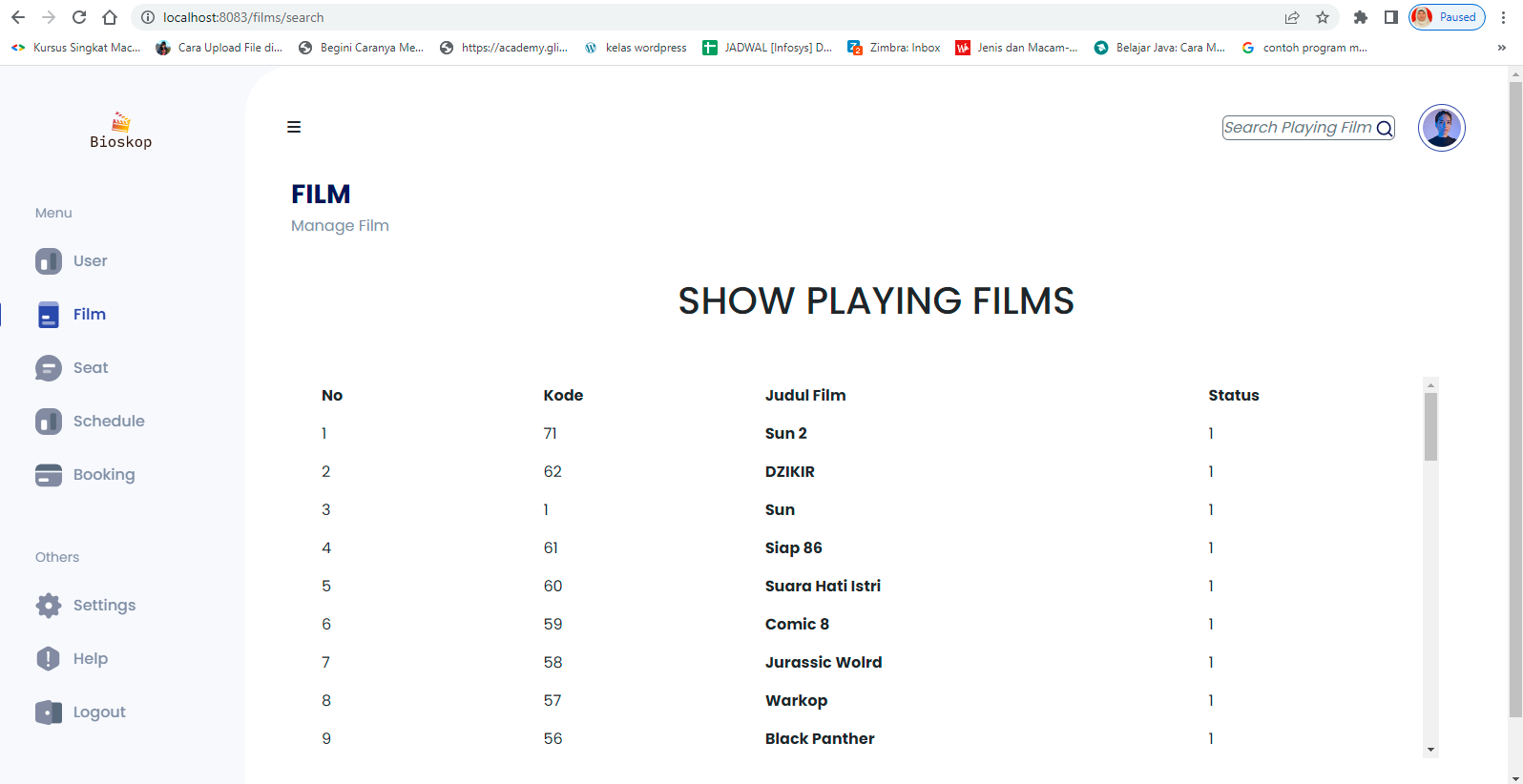
Code Description

200 OK

Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Film Id konsumen yang  berhasil di insert. | 62 |
| name | String | M |  | Nama film konsumen yang  berhasil di insert. | dzikir 24  hours |
| isPlaying | Integer | M |  | Status tayang film tersedia atau tidak yang berhasil di  insert. | 1 |

* 1. POST FILM IS PLAYING
* Tampilan Film untuk menampilkan film berdasarkan nomor, Kode, Judul Film, Status sedang tayang (1) atau status sedang tidak tayang (0).



* Endpoint

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/findFilmIsPlaying>

* End Point Front End :

<http://localhost:8083/films/search>

* Request URL

<http://localhost:8083/swagger-ui/index.html#/films-controller-dto/findFilmIsPlaying>

Sample Request

|  |
| --- |
|  |

* Sample Response

|  |
| --- |
| **Succes Responce** |
| {  “data”: [  {  “filmId” : 2,  “name” : The Roundup”,  “isPlaying : 1  }  ],  “message”: “Find Film Succes”,  “status” : 200 |

{

"data": "Failed create data",

"message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",

"status": 400

}

Failed Response

* Reaquest Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Film Id konsumen yang | 62 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | ingin di mainkan |  |
| name | String | M |  | Nama film konsumen yang  ingin di mainkan | dzikir 24  hours |
| isPlaying | Integer | M |  | Status tayang film tersedia atau  tidak | 1 |

* Respon body

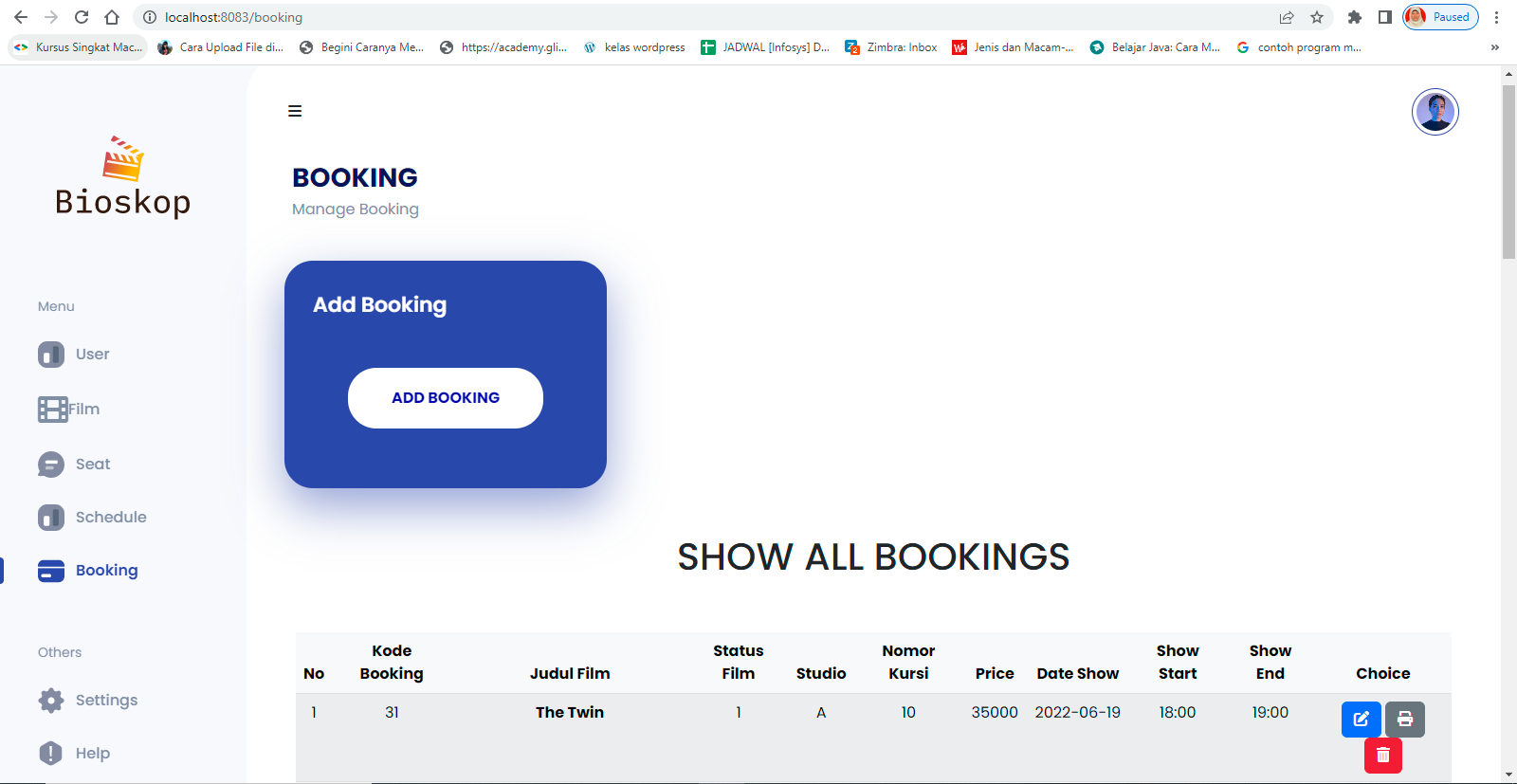
|  |
| --- |
| Respon Body |
| {  “data”: [  {  “FilmId”:2  “name”: "The Roundoup”  “isPlaying”:1  }, |

* Response structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| filmId | Long | M |  | Film Id konsumen yang  berhasil di insert. | 62 |
| name | String | M |  | Nama film konsumen yang  berhasil di insert. | dzikir 24  hours |
| isPlaying | Integer | M |  | Status tayang film tersedia atau tidak yang berhasil di  insert. | 1 |

1. BOOKING
   1. GET ALL

* Tampilan Booking untuk menampilkan reservasi tiket berdasarkan kode booking, judul film, status film sedang tayang (1) atau status film sedang tidak tayang(0),studio, nomor kursi, price, data show, show start, show end dan tombol create booking, edit serta delete.



* End Point :

http://localhost:8083/swagger-ui/index.html#/booking-controller/getBookingById

* End Point Front End:

http://localhost:8083/booking

* Request URL

http://localhost:8083/swagger-ui/index.html#/booking-controller/getBookingById

* Sample Response Request

|  |  |
| --- | --- |
| Success Response | |
| { |  |
|  | "data": [ |
|  | "book\_id": 9, |
|  | "usr\_id": 1, |
|  | "usr\_name": "Wahyu Wijanarko", |
|  | "email\_usr": ["WahyuW@gmail.com"](mailto:WahyuW@gmail.com), |
|  | "film\_name": "Jurassic World Dominion", |
|  | "price": 35000, |
|  | "studio": "A", |
|  | "status\_show": 1, |
|  | "seat\_num": 1, |
|  | "status\_seat": 1, |
|  | "date\_film": "2022-06-19", |
|  | "start\_film": "18:00:00", |
|  | "end\_film": "19:00:00" |
|  | } |
|  | ], |
|  | "message": "Success Get All", |
|  | "status": 200 |
| } |  |

}

"data": "Table has no value",

"message": "Data Booking not exist", "status": 404

{

Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| book\_id | Integer | M |  | Nomor Identifikasi booking  dari system. | 1 |
| usr\_id | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| usr\_name | String | M |  | Nama user | Budima  n Piatu |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| email\_us r | Stirng | M |  | Email User | Budima nPiatu@ gmail.co  m |
| film\_nam e | String | M |  | Nama film | Jurassic World  Dominat  ion |
| price | Integer | M |  | Harga tiket film | 20000 |
| studio | String | M |  | Nama Studio | A |
| status\_sh  ow | Integer | M |  | Film tayang (1) atau tidak (0) | 1 |
| seat\_num | Integer | M |  | Nomor kursi | 69 |
| status\_se  at | Integer | M |  | Ketersedian kursi (1) atau  tidak (0) | 1 |
| date\_film | date | M |  | Tanggal Tayang Film | 2022-06  -19 |
| start\_film | time | M |  | Waktu Mulai Film | 18:00:00 |
| end\_film | time | M |  | Waktu Selesai Film | 19:00:00 |

* 1. GET BY ID
* End Point :

http://localhost:8083/swagger-ui/index.html#/

* Request URL

http://localhost:8083/swagger-ui/index.html#/

* Sample Response Request

"data": {

{

Success Response

"book\_id": 4,

"usr\_id": 4,

"usr\_name": "Nur Aida",

"email\_usr": ["Aida\_Nur@gmail.com"](mailto:Aida_Nur@gmail.com),

"film\_name": "di bawah lindungan kabah", "price": 35000,

"studio": "B",

"status\_show": 0,

"seat\_num": 2,

"status\_seat": 1,

"date\_film": "2022-08-08",

"start\_film": "00:10:00",

"end\_film": "00:15:00"

},

"message": "Success Get By Id", "status": 200

}

}

"data": "Data not found",

"message": "Booking not exist with id 15", "status": 404

{

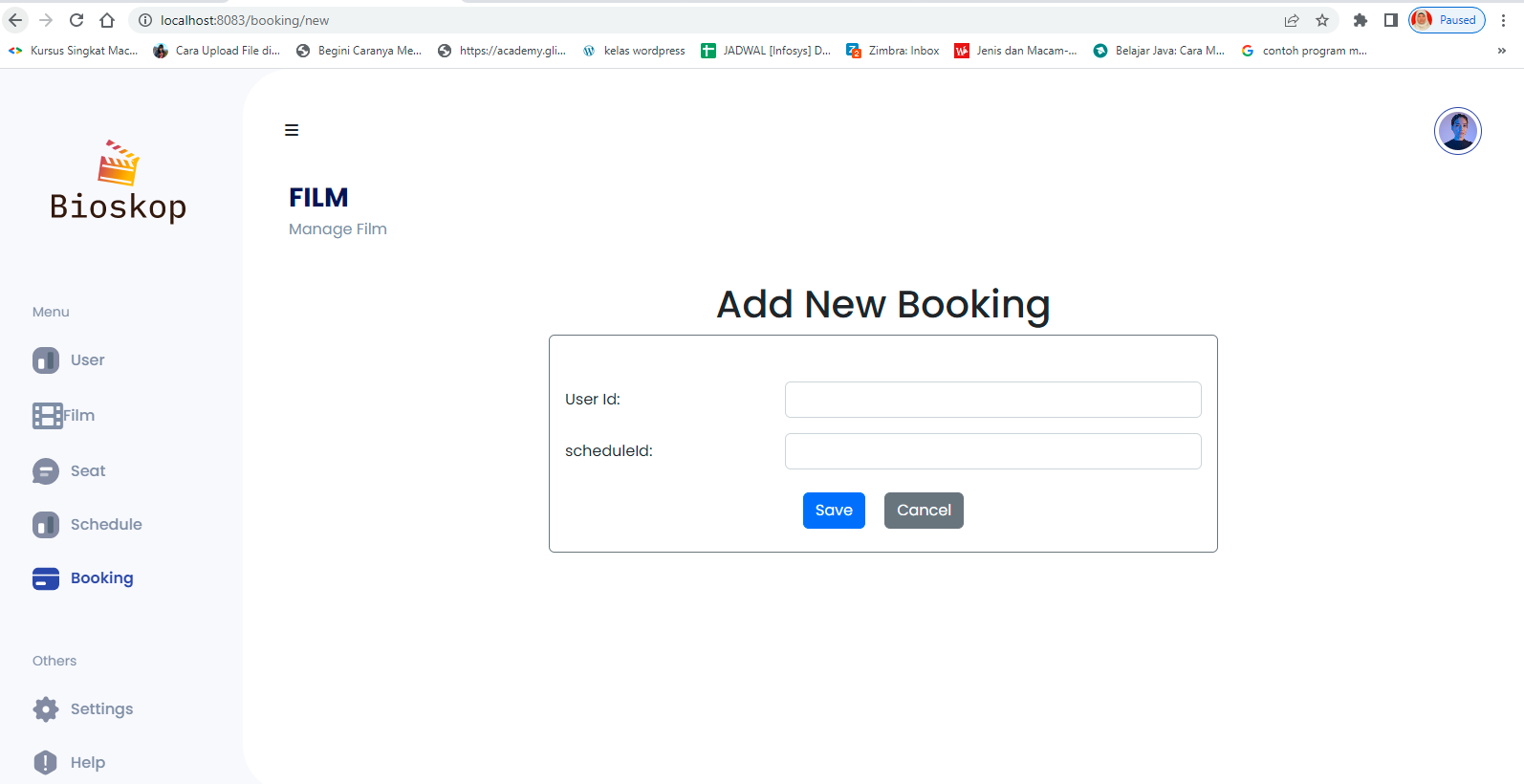
Failed Response

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandator  y | Length | Description | Example |
| book\_id | Integer | M |  | Nomor Identifikasi booking  dari system. | 1 |
| usr\_id | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| usr\_name | String | M |  | Nama user | Budiman Piatu |
| email\_us  r | Stirng | M |  | Email User | Budiman  Piatu@g mail.com |
| film\_nam  e | String | M |  | Nama film | Jurassic  World |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Dominati  on |
| price | Integer | M |  | Harga tiket film | 20000 |
| studio | String | M |  | Nama Studio | A |
| status\_sh  ow | Integer | M |  | Film tayang (1) atau tidak (0) | 1 |
| seat\_num | Integer | M |  | Nomor kursi | 69 |
| status\_se  at | Integer | M |  | Ketersedian kursi (1) atau  tidak (0) | 1 |
| date\_film | date | M |  | Tanggal Tayang Film | 2022-06-  19 |
| start\_film | time | M |  | Waktu Mulai Film | 18:00:00 |
| end\_film | time | M |  | Waktu Selesai Film | 19:00:00 |

* 1. CREATE/POST
* Tampilan add new booking digunakan untuk menambahkan film baru berdasarkan userId dan scheduleid.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/booking-controller/bookingCreate>

* End Point Frond End:

http://localhost:8083/booking/new

* Request URL

http://localhost:8083/swagger-ui/index.html#/booking-controller/bookingCreate

* Sample Request

{

"usr": {

"userId":15

},

"sch": {

"scheduleId": 1

}

}

* Sample Response Request

"data"

"book\_id" 15

"sch\_id" 1

{

Success Response

"usr\_id" 15

"message" "Success Create Booking" "status" 201

}

}

"data": "Data not found",

"message": "Booking not exist with id 15", "status": 404

{

Failed Response

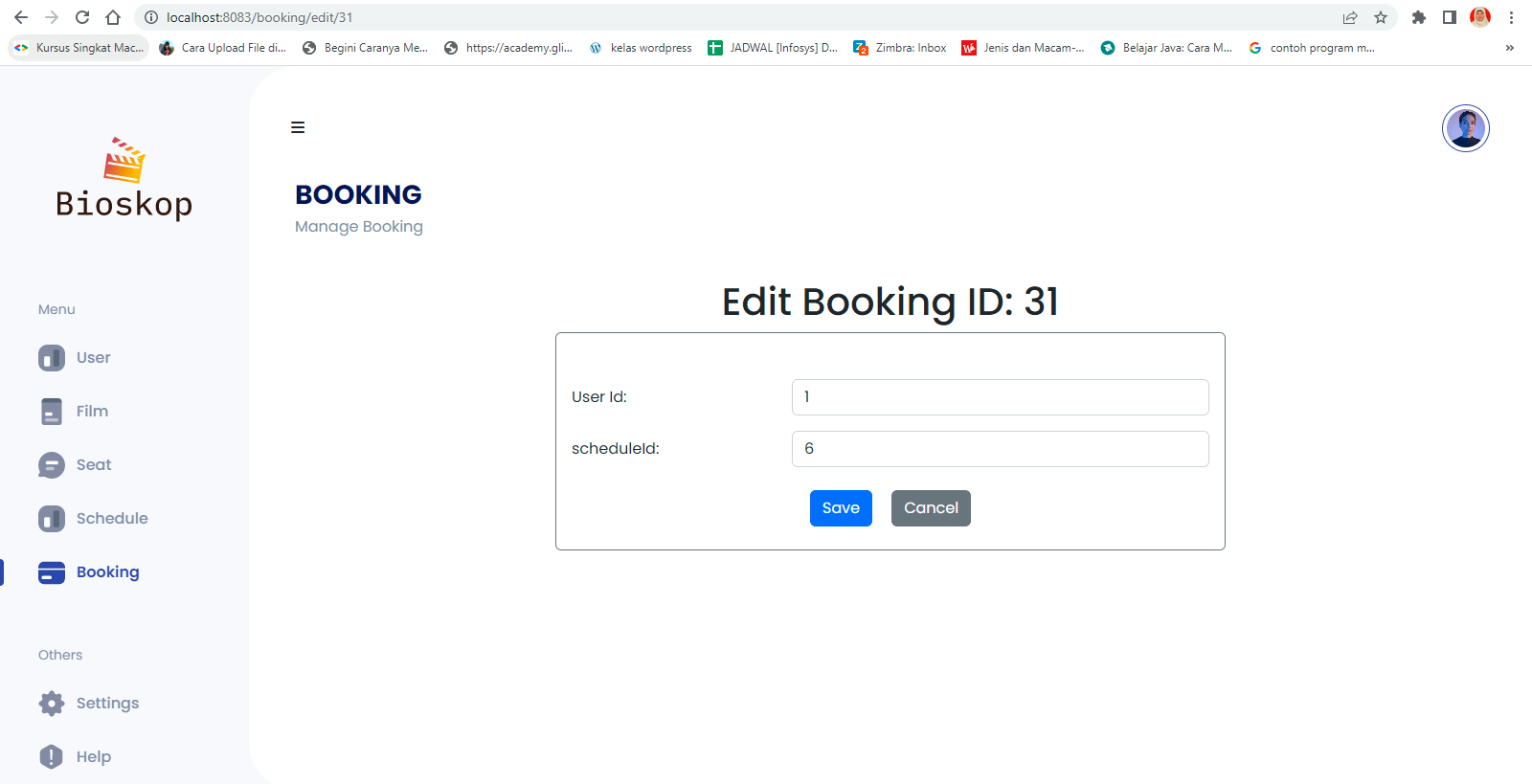
* Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| Usr | Entity/Mod  el | M |  | Enitity.User |  |
| userId | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| Sch | Entity/Mod  el | M |  | Entity Schedule |  |
| schedul  eId | Stirng | M |  | Nomor Identifikasi schedule  dari System | 1 |

* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| book\_id |  | M |  | Nomor Identifikasi booking  dari system. | 11 |
| sch\_id | Integer | M |  | Nomor Identifikasi schedule  dari system. | 12 |
| usr\_id | Integer | M |  | Nomor Identifikasi user dari  system. | 13 |

* 1. UPDATE/PUT
* Tampilan edit booking digunakan untuk merubah data yang tersimpan berdasarkan userId dan scheduleid.



* End Point :

localhost:8083/swagger-ui/index.html#/booking-controller/bookingupdate

* End Point Front End :

localhost:8083/swagger-ui/index.html#/booking-controller/bookingupdate

http://localhost:8083/booking/edit/31

* Request URL
* Sample Request

{

"usr": {

"userId":15

},

"sch": {

"scheduleId": 1

}

}

* Sample Response Request

|  |  |  |
| --- | --- | --- |
| Success Response | | |
|  | { |  |
|  |  | "data" |
|  |  | "book\_id" 1 |
|  |  | "usr\_id" 15 |
|  |  | "usr\_name" " Taji Rangkang” |
|  |  | "email\_usr" [" Taji Rangkang@apps.ipb.ac.id "](mailto:%22%20Taji%20Rangkang@apps.ipb.ac.id%20%22) |
|  |  | "film\_name" "Sun" |
|  |  | "price" 35000 |
|  |  | "studio" "A" |
|  |  | "status\_show" 1 |
|  |  | "seat\_num" 1 |
|  |  | "status\_seat" 1 |
|  |  | "date\_film" "2022-06-19" |
|  |  | "start\_film" "18:00:00" |
|  |  | "end\_film" "19:00:00" |
|  |  | "message" "Success Update Booking" |
|  |  | "status" 201 |
| } |  |  |

|  |  |
| --- | --- |
| Failed Response | |
| { |  |
|  | "data" "Bad Request" |
|  | "message" "Booking not exist with id 200" |
|  | "status" 400 |
| } |  |

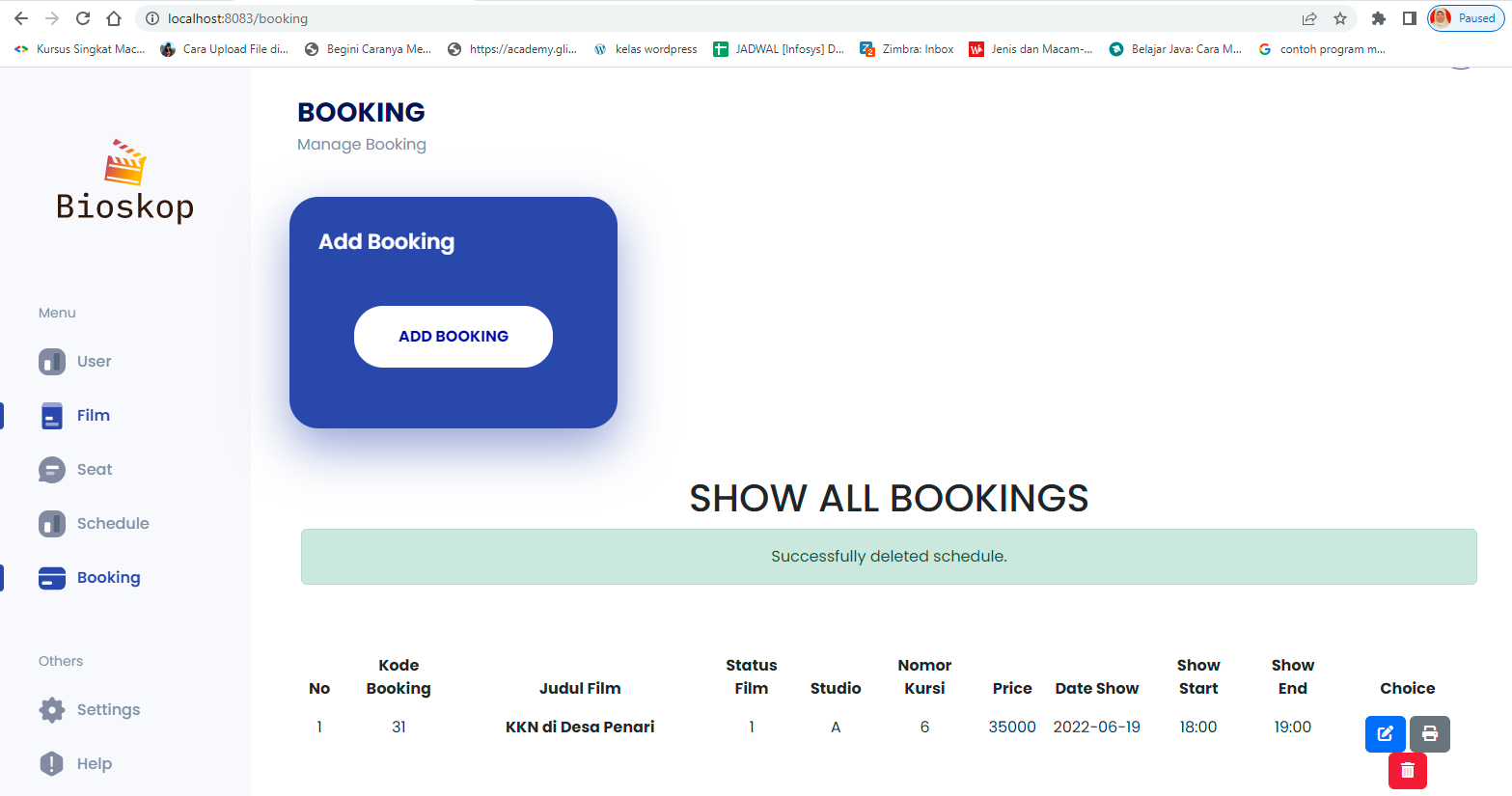
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Exampl  e |
| Usr | Entity/Mod  el | M |  | Enitity.User |  |
| userId | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| Sch | Entity/Mod  el | M |  | Entity Schedule |  |
| schedule  Id | Stirng | M |  | Nomor Identifikasi schedule  dari System | 1 |

* Request Structure
* Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandatory | Length | Description | Example |
| book\_id | Integer | M |  | Nomor Identifikasi booking  dari system. | 1 |
| usr\_id | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| usr\_na  me | String | M |  | Nama user | Budima n Piatu |
| email\_u sr | Stirng | M |  | Email User | Budima nPiatu@ gmail.co  m |
| film\_na me | String | M |  | Nama film | Jurassic World  Dominat  ion |
| price | Integer | M |  | Harga tiket film | 20000 |
| studio | String | M |  | Nama Studio | A |
| status\_s  how | Integer | M |  | Film tayang (1) atau tidak (0) | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| seat\_nu  m | Integer | M |  | Nomor kursi | 69 |
| status\_s  eat | Integer | M |  | Ketersedian kursi (1) atau  tidak (0) | 1 |
| date\_fil  m | date | M |  | Tanggal Tayang Film | 2022-06  -19 |
| start\_fil  m | time | M |  | Waktu Mulai Film | 18:00:00 |
| end\_fil  m | time | M |  | Waktu Selesai Film | 19:00:00 |

* 1. DELETE
* Tampilan delete untuk menghapus dari data yang tersimpan seperti kode booking, judul film, status film sedang tayang (1) atau status film tidak tayang(0)atau status 0 (sedang tidak tayang), studio, nomor kursi, price, data show, show start, show end.



* End Point :

<http://localhost:8083/swagger-ui/index.html#/booking-controller/deleteBooking_1>

* End Point Front End:

http://localhost:8083/booking

* Request URL

http://localhost:8083/swagger-ui/index.html#/booking-controller/deleteBooking\_1

* Sample Response Request

}

"data": **true**,

"message": "Success Delete Booking by ID", "status": 200

{

Success Response

}

"data": "Data not found",

"message": "Booking not exist with id 100", "status": 404

{

Failed Response

* 1. POST/QUERY BY FILMNAME
* End Point :

http://localhost:8083/swagger-ui/index.html#/booking-controller/findBookingBySchdeuleFilmName

* Request URL

http://localhost:8083/swagger-ui/index.html#/booking-controller/findBookingBySchdeuleFilmName

* Sample Request

{

"name": "Avatar"

}

* Sample Response Request

}

"data": [

{

"book\_id": 2,

"usr\_id": 6,

"usr\_name": "Lionel",

"email\_usr": ["Lionel@gmail.com"](mailto:Lionel@gmail.com), "film\_name": "Avatar 2",

"price": 35000, "studio": "A",

"status\_show": 1,

"seat\_num": 3,

"status\_seat": 1,

"date\_film": "2022-06-19",

"start\_film": "18:00:00",

"end\_film": "19:00:00"

}

],

"message": "Succes Query By Filmname", "status": 200

{

Success Response

}

"data": "Failed Query By Filename",

"message": "Booking not exist with Filmname cinta itu nyata", "status": 400

{

Failed Response

* Request Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandator  y | Length | Description | Example |
| name | String | M |  | Filmname | Avatar |

* + - * Response Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Data Type | Mandator  y | Length | Description | Example |
| book\_i  d | Integer | M |  | Nomor Identifikasi booking  dari system. | 1 |
| usr\_id | Integer | M |  | Nomor Identifikasi user dari  system. | 1 |
| usr\_na  me | String | M |  | Nama user | Budiman Piatu |
| email\_ usr | Stirng | M |  | Email User | Budiman Piatu@g  mail.com |
| film\_na me | String | M |  | Nama film | Jurassic World  Dominati  on |
| price | Integer | M |  | Harga tiket film | 20000 |
| studio | String | M |  | Nama Studio | A |
| status\_s  how | Integer | M |  | Film tayang (1) atau tidak  (0) | 1 |
| seat\_nu  m | Integer | M |  | Nomor kursi | 69 |
| status\_s  eat | Integer | M |  | Ketersedian kursi (1) atau  tidak (0) | 1 |
| date\_fil  m | date | M |  | Tanggal Tayang Film | 2022-06-  19 |
| start\_fil  m | time | M |  | Waktu Mulai Film | 18:00:00 |
| end\_fil | time | M |  | Waktu Selesai Film | 19:00:00 |
|  |  |  |  |  |  |

* 1. Cetak Tiket Reservasi Bioskop
* Tampilan Cetak Tiket Reservasi Bioskop untuk mencetak atau menyimpan file dalam bentuk pdf.
* End Point Front End :

http://localhost:8083/booking/detail/31

