

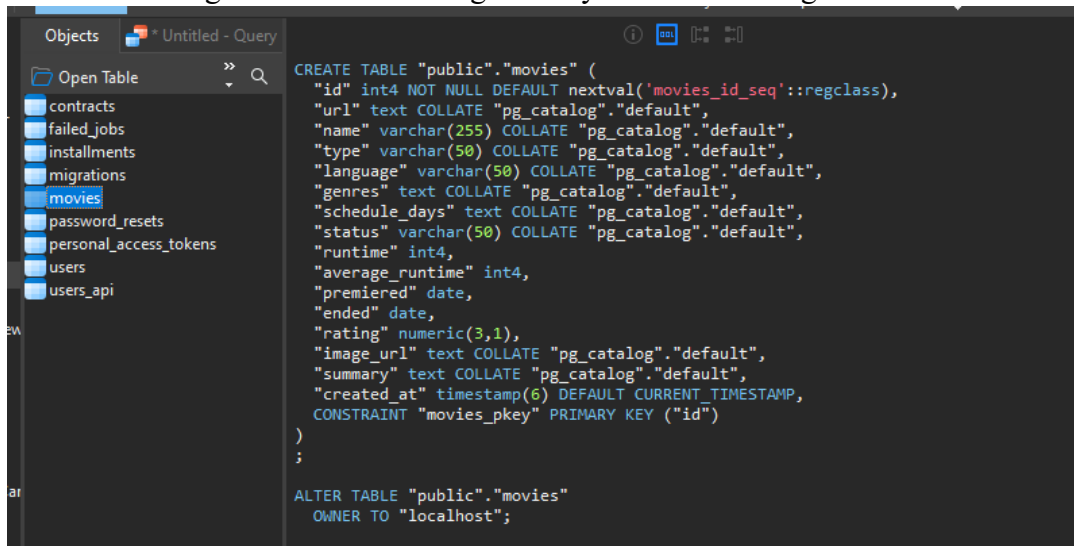
PT SIGMA CIPTA CARAKA CODING TEST

BY : RAJENDRA RAKHA ARYA PRABASWARA

POSITION : FULLSTACK DEVELOPER

GITHUB LINK CODE : [Rjndrkha/SigmaCipta](https://github.com/Rjndrkha/SigmaCipta)

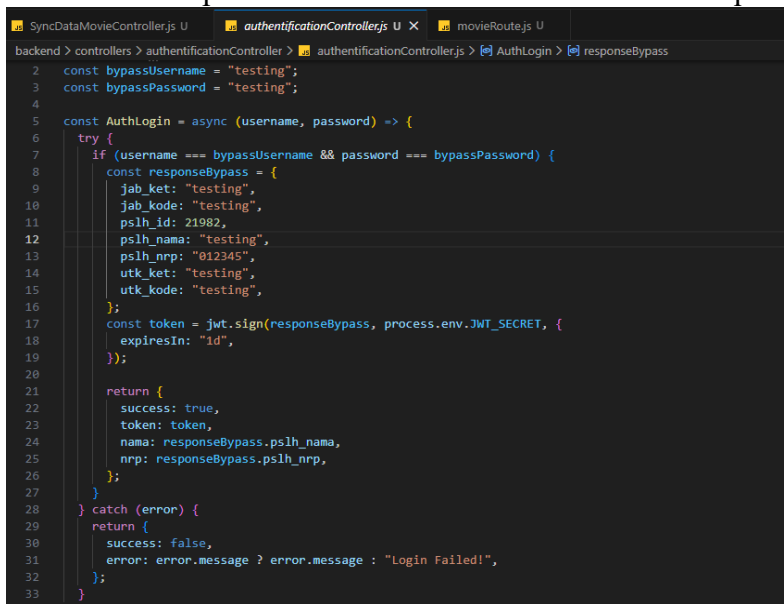
1. In Here Im using Framework React + Node + Tailwind
2. Database Configuration to store integration sync Here im using POSTGRE



The screenshot shows a database client interface with a sidebar on the left listing various database objects like 'contracts', 'failed_jobs', 'installments', 'migrations', 'movies', 'password_resets', 'personal_access_tokens', 'users', and 'users_api'. The main area displays the SQL code for creating a table named 'movies' in the 'public' schema. The table has columns for 'id', 'url', 'name', 'type', 'language', 'genres', 'schedule_days', 'status', 'runtime', 'average_runtime', 'premiered', 'ended', 'rating', 'image_url', 'summary', and 'created_at'. The 'id' column is the primary key. The 'created_at' column is a timestamp with a default value of 'CURRENT_TIMESTAMP'. The code also includes an 'ALTER TABLE' statement to change the owner of the 'movies' table to 'localhost'.

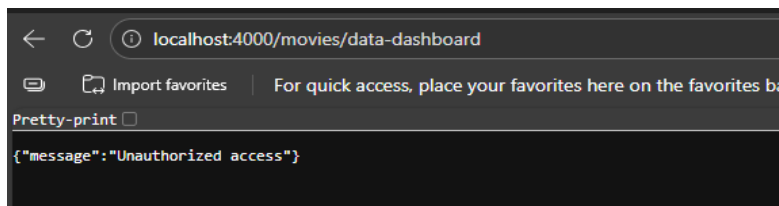
```
CREATE TABLE "public"."movies" (  
  "id" int4 NOT NULL DEFAULT nextval('movies_id_seq'::regclass),  
  "url" text COLLATE "pg_catalog"."default",  
  "name" varchar(255) COLLATE "pg_catalog"."default",  
  "type" varchar(50) COLLATE "pg_catalog"."default",  
  "language" varchar(50) COLLATE "pg_catalog"."default",  
  "genres" text COLLATE "pg_catalog"."default",  
  "schedule_days" text COLLATE "pg_catalog"."default",  
  "status" varchar(50) COLLATE "pg_catalog"."default",  
  "runtime" int4,  
  "average_runtime" int4,  
  "premiered" date,  
  "ended" date,  
  "rating" numeric(3,1),  
  "image_url" text COLLATE "pg_catalog"."default",  
  "summary" text COLLATE "pg_catalog"."default",  
  "created_at" timestamp(6) DEFAULT CURRENT_TIMESTAMP,  
  CONSTRAINT "movies_pkey" PRIMARY KEY ("id")  
)  
;  
  
ALTER TABLE "public"."movies"  
  OWNER TO "localhost";
```

3. Here Also I Implement JWT Token To Secure When Request Data



The screenshot shows a code editor with a file explorer on the left. The active file is 'authenticationController.js'. The code implements an 'AuthLogin' function that takes 'username' and 'password' as arguments. It first checks if the username and password match the 'bypassUsername' and 'bypassPassword' constants. If they match, it returns a 'responseBypass' object. Otherwise, it generates a JWT token using 'jwt.sign' and returns an object with 'success', 'token', 'nama', and 'nnp' properties. If the login fails, it returns an object with 'success' set to false and an 'error' message.

```
const bypassUsername = "testing";  
const bypassPassword = "testing";  
  
const AuthLogin = async (username, password) => {  
  try {  
    if (username === bypassUsername && password === bypassPassword) {  
      const responseBypass = {  
        jab_ket: "testing",  
        jab_kode: "testing",  
        pslh_id: 21982,  
        pslh_nama: "testing",  
        pslh_nrp: "012345",  
        utk_ket: "testing",  
        utk_kode: "testing",  
      };  
      const token = jwt.sign(responseBypass, process.env.JWT_SECRET, {  
        expiresIn: "1d",  
      });  
      return {  
        success: true,  
        token: token,  
        nama: responseBypass.pslh_nama,  
        nnp: responseBypass.pslh_nrp,  
      };  
    }  
  } catch (error) {  
    return {  
      success: false,  
      error: error.message ? error.message : "Login Failed!",  
    };  
  }  
}
```



The screenshot shows a web browser window with the address bar displaying 'localhost:4000/movies/data-dashboard'. The browser's developer tools are open, showing a 'Pretty-print' section with the JSON response: '{\"message\": \"Unauthorized access\"}'.

```
{\"message\": \"Unauthorized access\"}
```

4. API Documentation

GET localhost:4000/movies/sync-data : To SYNC Data

GET <http://localhost:4000/movies/data> : To GET Data

GET <http://localhost:4000/movies/data/:id> : To GET Detail Data

POST <http://localhost:4000/movies/create> : To CREATE Data

POST <http://localhost:4000/movies/update> : To UPDATE Data

DELETE <http://localhost:4000/movies/delete/:id> : To DELETE Data

GET localhost:4000/movies/data-dashboard : To GET Data Dashboard

5. Login Page

PT SIGMA CIPTA CARAKA
Please Login To Access Portal

Login Using Username/Password
test

Username
test

Password
Input password

Masuk

6. Table View

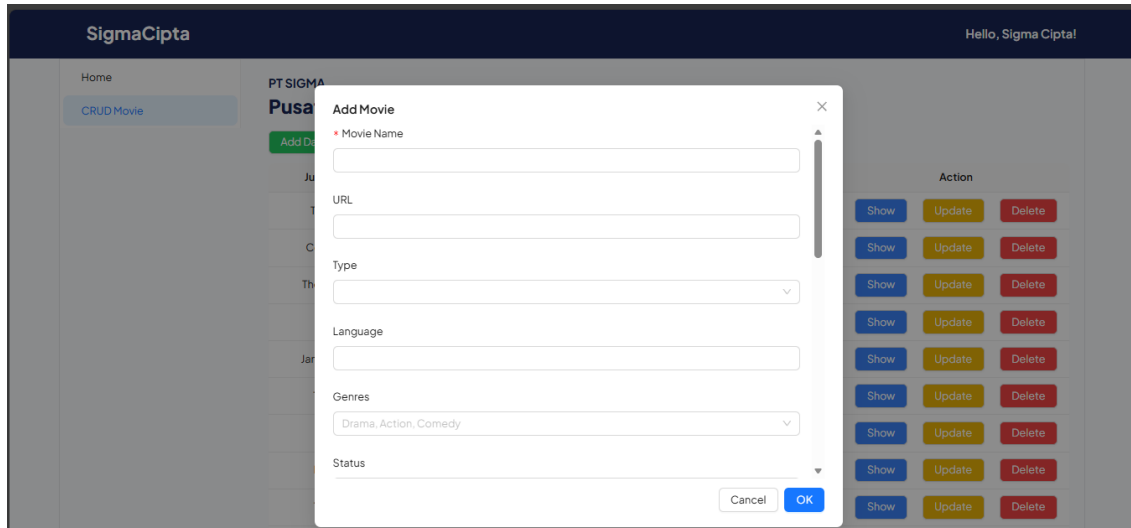
Judul Film	Genre	Runtime	Schedule	Last Sync Time	Action
Benched	Comedy, Legal	30	Tuesday	07 January 2026	Show Delete
Constantine	Drama, Action, Horror	60	Friday	07 January 2026	Show Delete
The Great Fire	Drama	60	Thursday	07 January 2026	Show Delete
Marry Me	Comedy	30	Tuesday	07 January 2026	Show Delete
Jane the Virgin	Drama, Crime, Romance	60	Wednesday	07 January 2026	Show Delete
The Affair	Drama, Romance	61	Sunday	07 January 2026	Show Delete
Cristela	Comedy, Legal	30	Friday	07 January 2026	Show Delete
Kingdom	Drama, Family, Sports	60	Wednesday	07 January 2026	Show Delete

Search average_runtime

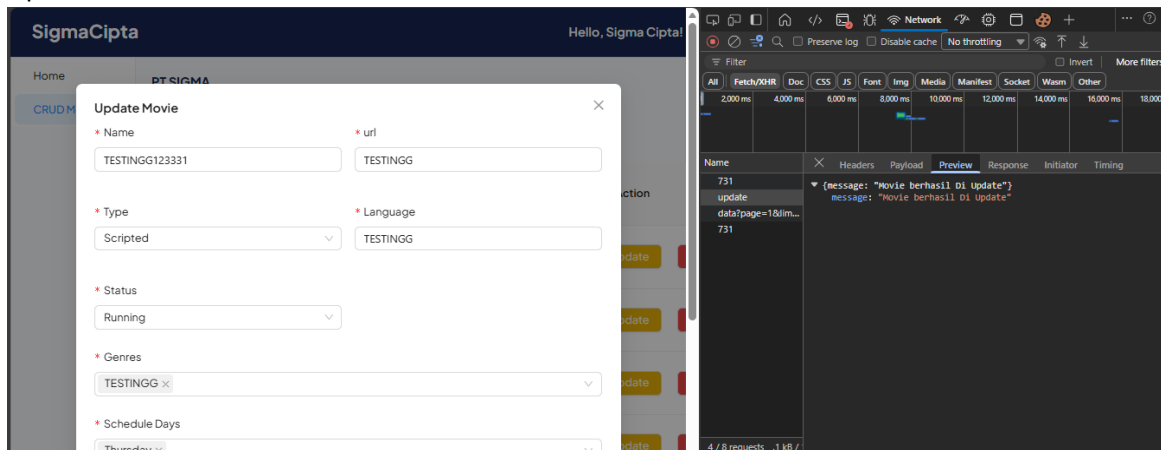
Name	Headers	Payload	Preview	Response	Initiator	Timing				
data?page=2&lim...		<div><div>▼ Query String Parameters</div><table><tr><td>page</td><td>2</td></tr><tr><td>limit</td><td>10</td></tr></table></div>	page	2	limit	10	<div>View source</div>		<div>View URL-encoded</div>	
page	2									
limit	10									
		<div><div>▼</div></div>								

The screenshot shows the SigmaCipta application interface. On the left, there's a sidebar with 'Home' and 'CRUD Movie' buttons. The main area displays a table of movies. A modal titled 'Movie Detail' is open, showing information for the movie 'Strange Empire'. The modal includes fields for Title, Poster, Premiered, Genre, Language, Status, and Summary. The background shows a table with columns for 'Sync Time' and 'Action' (Show/Delete).

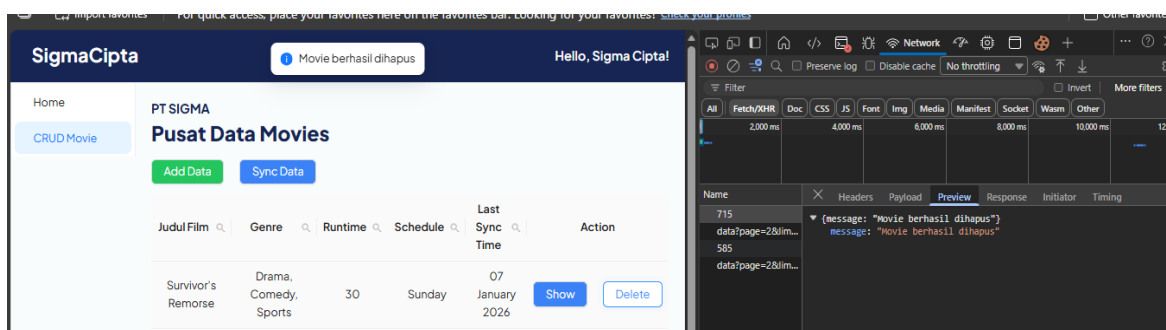
10. Add Movie



11. Update Movie



12. Delete Data



13. Sync Data

The screenshot shows the SigmaCipta application interface. On the left, a sidebar contains 'Home' and 'CRUD Movie'. The main content area is titled 'PT SIGMA Pusat Data Movies' and includes 'Add Data' and 'Sync Data' buttons. Below these is a table with columns: 'Judul Film', 'Genre', 'Runtime', 'Schedule', 'Last Sync Time', and 'Action'. The table contains one row with the value 'TESTINGG123331' in the 'Judul Film' column. To the right, a browser's developer tools network tab is open, showing a request to 'https://localhost:4000/movies/sync-data' with a status of '200 OK'.

14. Dashboard Movie

The screenshot shows the SigmaCipta application interface. On the left, a sidebar contains 'Home' and 'CRUD Movie'. The main content area is titled 'PT SIGMA Dashboard Data Movies'. It features a date range selector set to '2026-01-01' to '2026-01-08'. Below this is a calendar view for January and February 2026. To the right, there is a 'Latest Data' section listing movies: 'Strange Empire (Scripted)', 'Benched (Scripted)', 'State of Affairs (Scripted)', 'The McCarthys (Scripted)', and 'Mulaney (Scripted)'. Below the calendar, there is a 'Daily Movies Trend' section.

15. Chart

The screenshot shows the SigmaCipta application interface. On the left, a sidebar contains 'Home' and 'CRUD Movie'. The main content area is titled 'PT SIGMA Chart'. It features a 'Total Movies' section with the value '241' and a 'Most Category' section with the value 'Scripted'. Below these, there is a 'Movie By Genre' pie chart and a 'Daily Movies Trend' bar chart. The pie chart shows the distribution of movies by genre: Scripted (88.4%), Animation (6.9%), Reality (4.1%), Talk Show (5.8%), and Documentary (4.8%). The bar chart shows the daily movie trend over a period of four days.