1. Struktur prisma

Model Login {

Id Int @id @default(autoincrement())

Username String @unique

Password String

workLogs WorkLog[] // Relasi satu-ke-banyak dengan WorkLog

}

Model WorkLog {

Id Int @id @default(autoincrement())

Pic String

Week Int

Date DateTime

Opportunity String

Type String

Account String

Activity String

userId Int

user Login @relation(fields: [userId], references: [id]) // Relasi ke model Login

createdAt DateTime @default(now())

}

npx prisma migrate dev --name initial\_setup\_with\_relation

1. System login jwt

Npm install jsonwebtoken cookie

Login.jsx :

import { PrismaClient } from '@prisma/client';

import jwt from 'jsonwebtoken';

import cookie from 'cookie';

import bcrypt from 'bcryptjs';

const prisma = new PrismaClient();

export default async function handler(req, res) {

if (req.method === 'POST') {

const { username, password } = req.body;

try {

// Cek apakah user ada di database

const user = await prisma.login.findUnique({

where: { username },

});

if (!user) {

return res.status(401).json({ error: 'User not found' });

}

// Verifikasi password

const isValid = await bcrypt.compare(password, user.password);

if (!isValid) {

return res.status(401).json({ error: 'Invalid credentials' });

}

// Buat token JWT

const token = jwt.sign({ id: user.id, username: user.username }, process.env.JWT\_SECRET, { expiresIn: '1h' });

// Set token di cookies

res.setHeader('Set-Cookie', cookie.serialize('token', token, {

httpOnly: true,

secure: process.env.NODE\_ENV === 'production',

maxAge: 3600,

path: '/'

}));

res.status(200).json({ message: 'Login successful' });

} catch (error) {

res.status(500).json({ error: 'Login failed' });

}

} else {

res.status(405).json({ error: 'Method not allowed' });

}

}

1. API routing worklog
2. Add :

import { PrismaClient } from '@prisma/client';

import jwt from 'jsonwebtoken';

import cookie from 'cookie';

const prisma = new PrismaClient();

export default async function handler(req, res) {

if (req.method === 'POST') {

try {

// Ambil token JWT dari cookies

const { token } = cookie.parse(req.headers.cookie || '');

// Verifikasi token JWT dan dapatkan userId

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

const userId = decoded.id;

const { pic, week, date, opportunity, type, account, activity } = req.body;

// Buat work log baru

const worklog = await prisma.workLog.create({

data: {

pic,

week: parseInt(week),

date: new Date(date),

opportunity,

type,

account,

activity,

userId, // Simpan userId dari token

},

});

res.status(200).json(worklog);

} catch (error) {

res.status(401).json({ error: 'Unauthorized' });

}

} else {

res.status(405).json({ error: 'Method not allowed' });

}

}

1. Get :

import { PrismaClient } from '@prisma/client';

import jwt from 'jsonwebtoken';

import cookie from 'cookie';

const prisma = new PrismaClient();

export default async function handler(req, res) {

if (req.method === 'GET') {

try {

// Ambil token JWT dari cookies

const { token } = cookie.parse(req.headers.cookie || '');

// Verifikasi token JWT dan dapatkan userId

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

const userId = decoded.id;

// Hanya ambil work log yang sesuai dengan userId

const workLogs = await prisma.workLog.findMany({

where: {

userId,

},

});

res.status(200).json(workLogs);

} catch (error) {

res.status(401).json({ error: 'Unauthorized' });

}

} else {

res.status(405).json({ error: 'Method not allowed' });

}

}

1. Update

// Mirip dengan add.js tapi dengan metode update

Import { PrismaClient } from ‘@prisma/client’;

Import jwt from ‘jsonwebtoken’;

Import cookie from ‘cookie’;

Const prisma = new PrismaClient();

Export default async function handler(req, res) {

If (req.method === ‘PUT’) {

Try {

Const { token } = cookie.parse(req.headers.cookie || ‘’);

Const decoded = jwt.verify(token, process.env.JWT\_SECRET);

Const userId = decoded.id;

Const { id, pic, week, date, opportunity, type, account, activity } = req.body;

Const worklog = await prisma.workLog.update({

Where: { id: parseInt(id) },

Data: {

Pic,

Week: parseInt(week),

Date: new Date(date),

Opportunity,

Type,

Account,

Activity,

},

});

Res.status(200).json(worklog);

} catch (error) {

Res.status(401).json({ error: ‘Unauthorized’ });

}

} else {

Res.status(405).json({ error: ‘Method not allowed’ });

}

}

1. Delete

Import { PrismaClient } from ‘@prisma/client’;

Import jwt from ‘jsonwebtoken’;

Import cookie from ‘cookie’;

Const prisma = new PrismaClient();

Export default async function handler(req, res) {

If (req.method === ‘DELETE’) {

Try {

Const { token } = cookie.parse(req.headers.cookie || ‘’);

Const decoded = jwt.verify(token, process.env.JWT\_SECRET);

Const userId = decoded.id;

Const { id } = req.body;

Await prisma.workLog.delete({

Where: { id: parseInt(id) },

});

Res.status(200).json({ message: ‘Work log deleted’ });

} catch (error) {

Res.status(401).json({ error: ‘Unauthorized’ });

}

} else {

Res.status(405).json({ error: ‘Method not allowed’ });

}

}

4.Front end work log

Worklog.jsx :

Import React, { useState, useEffect } from ‘react’;

Import Add from ‘../component/Add\_Log.jsx’;

Import Update from ‘../component/Update\_Log.jsx’;

Import Delete from ‘../component/Delete.jsx’;

Const WorkLog = () => {

Const [workLogs, setWorkLogs] = useState([]);

// Fetch data dari API saat komponen dimount

useEffect(() => {

const fetchWorkLogs = async () => {

try {

const response = await fetch(‘/api/worklog/get’);

const data = await response.json();

setWorkLogs(data);

} catch (error) {

Console.error(‘Error fetching work logs:’, error);

}

};

fetchWorkLogs();

}, []);

Return (

<div>

<h3>Work Log</h3>

<table>

<thead>

<tr>

<th>No</th>

<th>PIC</th>

<th>Week</th>

<th>Date</th>

<th>Opportunity</th>

<th>Type</th>

<th>Account</th>

<th>Activity</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{workLogs.map((log, index) => (

<tr key={log.id}>

<td>{index + 1}</td>

<td>{log.pic}</td>

<td>{log.week}</td>

<td>{new Date(log.date).toLocaleDateString()}</td>

<td>{log.opportunity}</td>

<td>{log.type}</td>

<td>{log.account}</td>

<td>{log.activity}</td>

<td>

{/\* Action buttons for Update and Delete \*/}

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

Export default WorkLog;