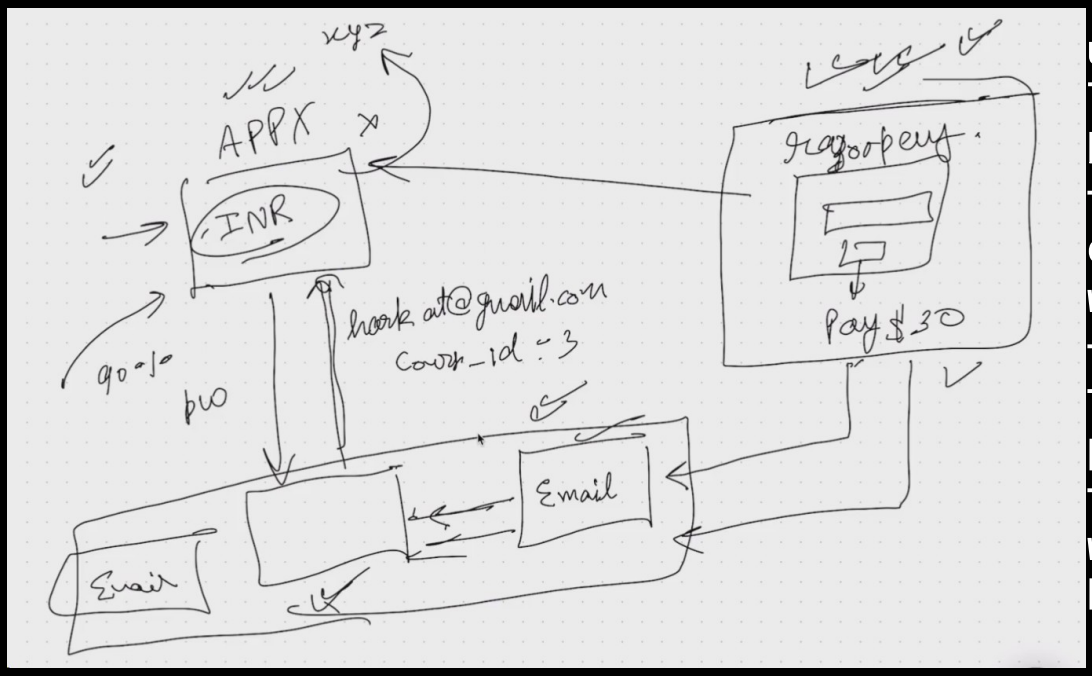
ZAPIER PROJECT

Use of zapier
razorpay = payment gateway



usecase of zapier like when you do buy a course there are many options imagine we bought service from appx and they just provide payment in inr so the issue is we want in usd also how we will do?

To fix that we like create our own razorpay page in which we redirect and accept the payment and after payment is received we send them email and then the we create a zapier flow like whenever a payment is received the email and course id will be send to the backend of appx to register this user and this will create a id pass and will send to user

HERE THE ZAPIER HELPED IN CONNECTING THE TWO BACKEND WHICH HELPED IN AUTOMATING THE FLOW THIS IS HOW WE FIXED IT

Another usecase like whenever I complete a recording it automatically saves and transfer to db and then show in the course page

What is a webhook?
This helps is contacting the backend of two different companies together

how does the payment gateway works??
like for any course website the banks cannot give api to connect to the app so what these stripe, razorpay does is like they contact with bank and they get the api key to work with and then we use that to make payments

so whenever a person does a payment he is redirected to bank page and then after a successful payment the bank tells the razorpay that the payment is done and the connection made between this two is called as the webhook which does this
after this only the razorpay also sends to course backend now you can give access to this user

and imagine you are thinking like if I just get the endpoint which hits the backend from the razor pay then I can just register my self that is not possible because when we create a razorpay account they also give like a secret which is passed with every successful transaction which is also verified

ANOTHER USECASE like when harkirat write in github like /bounty $10 then there we can also set like zapier workflow which will lead to like a email sent to that person and the money will be received using solana or some wallet

or the above usecase can also be done using webhooks get a url for the usecase from the zapier and then paste that in github repo inside the github webhooks and whenever you write the command it will automatically trigger that

like for this /bounty $10 --> triggers the zappier --> Mail sent --> added in the notion doc also

STARTING WITH THE CODE

creating the folder for hook
npm init -y
npx tsc --init

"type": "module" add this in package.json also

uncomment the src and rootdir

creating the src folder which contains index.ts

npm install express @types/express

first getting the userid and zapid from the url using req.headers.params

intializing prisma

npm install prisma

npx prisma init

@id to make the field primary

```
model Trigger{
    id          String      @id @default(uuid())
    zapId       String
    type        AvailabeTriggers @relation(fields:[zapId],references:[id])
}

model AvailabeTriggers{
    id          String      @id @default(uuid())
    name        String
    triggers    Trigger[]
}
```

both of them are in a relationship and are dependent on each other

one to many relationship

one to one relationship

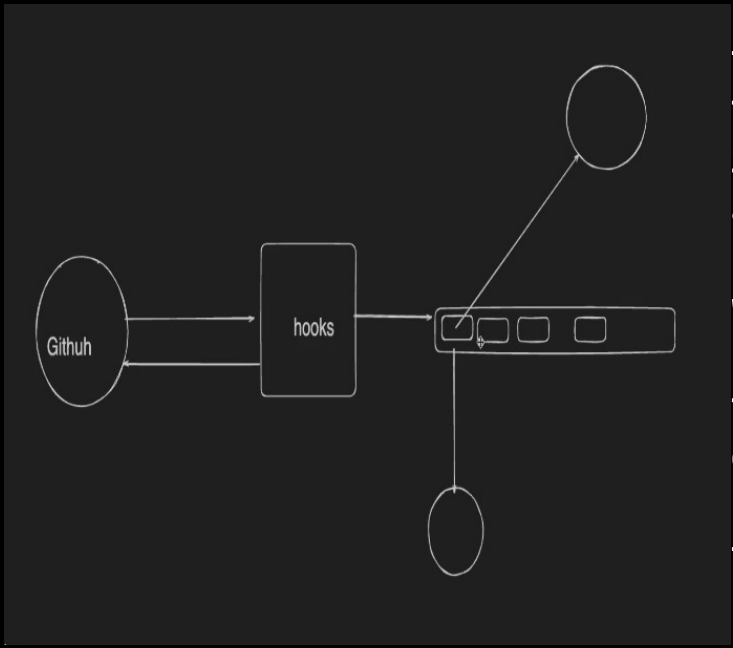
```
model Zap{
    id          String      @id @default(uuid())
    triggerId   String
    trigger     Trigger?
}

model Trigger{
    id          String      @id @default(uuid())
    zapId       String      @unique
    triggerId   String
    type        AvailabeTriggers @relation(fields:[triggerId],references:[id])
    zap         Zap              @relation(fields:[zapId],references: [id])
}

model AvailabeTriggers{
    id          String      @id @default(uuid())
    name        String
    triggers    Trigger[]
}
```

between the zap and trigger the relationship of one to one

YOU REMEMBER THE TRANSACTION PART WHICH IS VERY IMPORTANT IN THE PAYTM PROJECT

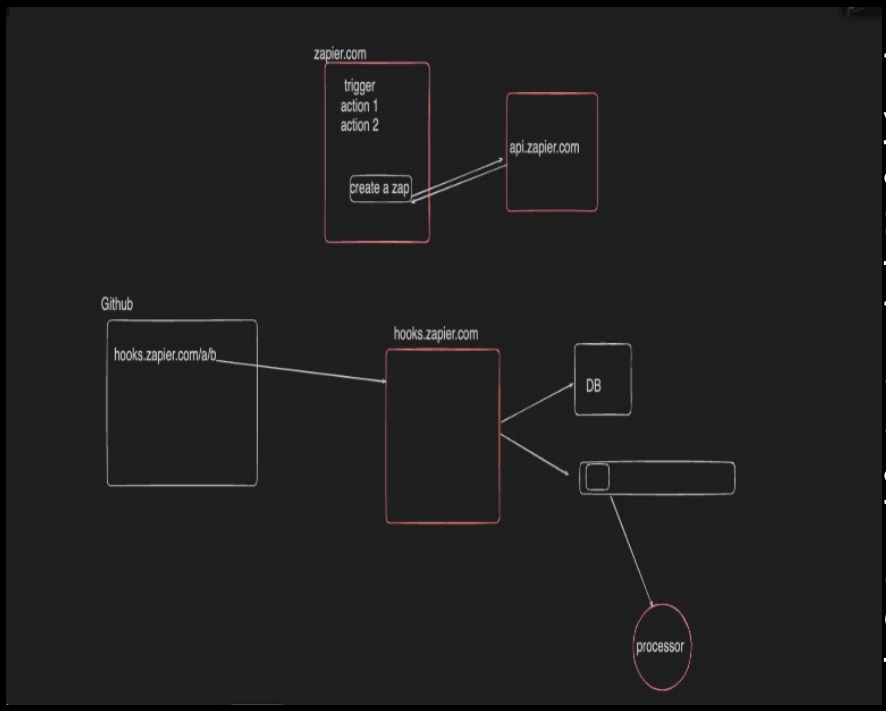
LIKE WHENVER A TRANSACTION HAPPENS IT SHOULD BE LIKE FULLY HAPPEN OR NOTHING SHOULD HAPPEN FOR THAT WE USE THE TRANSACTION OF DB AND NEVER A PARTIAL THING WILL HAPPEN

APPROACHES FOR HOOKS TO BE DELIVERED



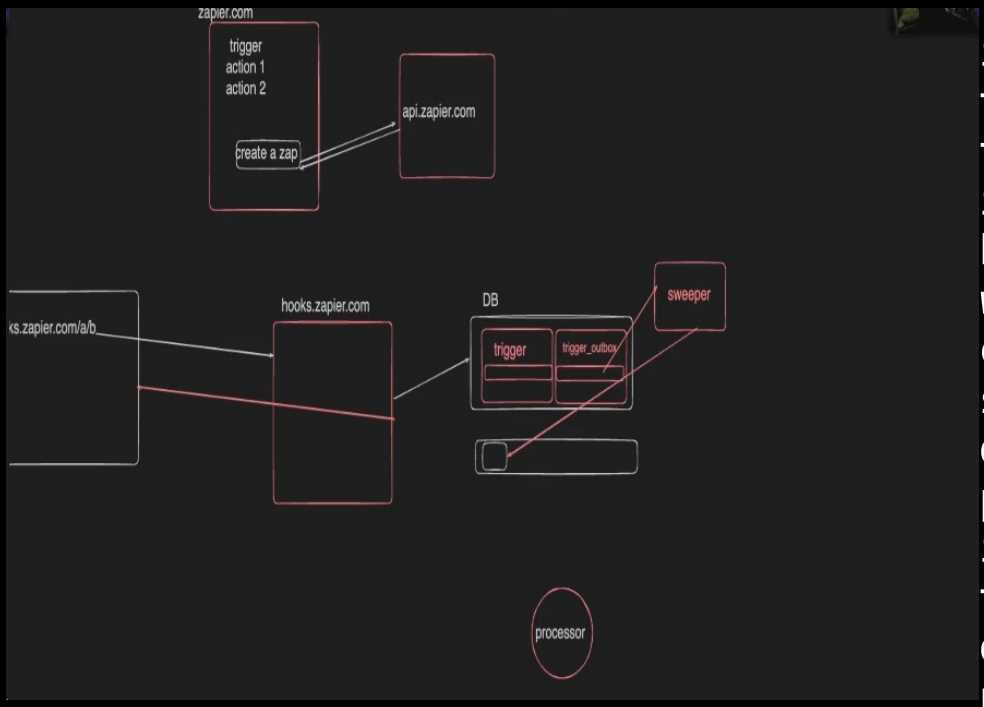
this is simply like whenver there is a event the hook will register and then will put in a ready queue in the kafka server and then the github will then directly show the hooks is executed , then from the ready queue the db and the db which process the data will use the data and will execute the action on this specific zap but issue is sometime in github it will show that the work is done but in zapier it will show that nothing is done which means the db server has not picked the data yet

HOW ZAPIER WORKS??



like first on the website you attach a trigger, actions and then create a zap which gets stored in the db and then returns a link, that link is then stored in the github repo settings in the webhook and we tick the settings like when we comment it should trigger the command and send the data then its sent and then stored in the kafka, redis queue and then its processed by the processor and stored in the db also

THIS APPROACH IS TRANSACTION OUTBOX PATTERN
THIS INVOLVES LIKE atomicity concept

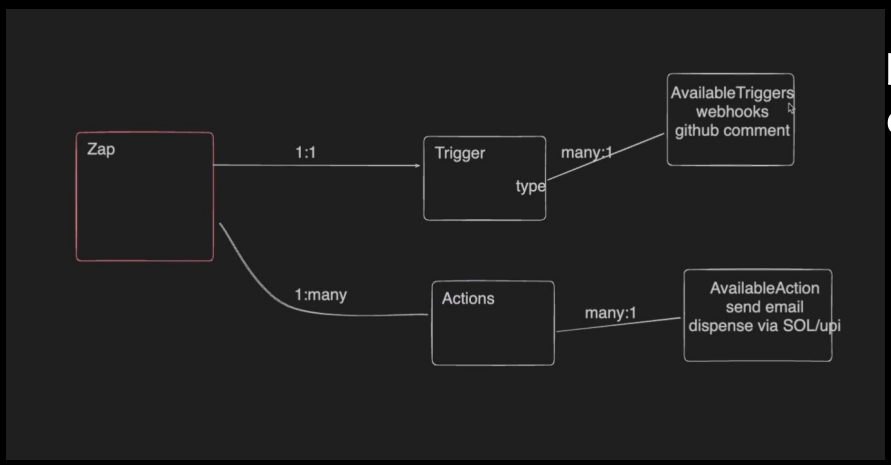


in this there is a transaction inside the DB which happens if both the things happens then only it will proceed otherwise it will shuts the process down and whenver the process is successful it will tell the user that the work will be done and it also passed to processor using sweeper

each trigger is a type of available trigger(google calender, web hooks)
every action is also a type of available action(sending a email, sending bounty)

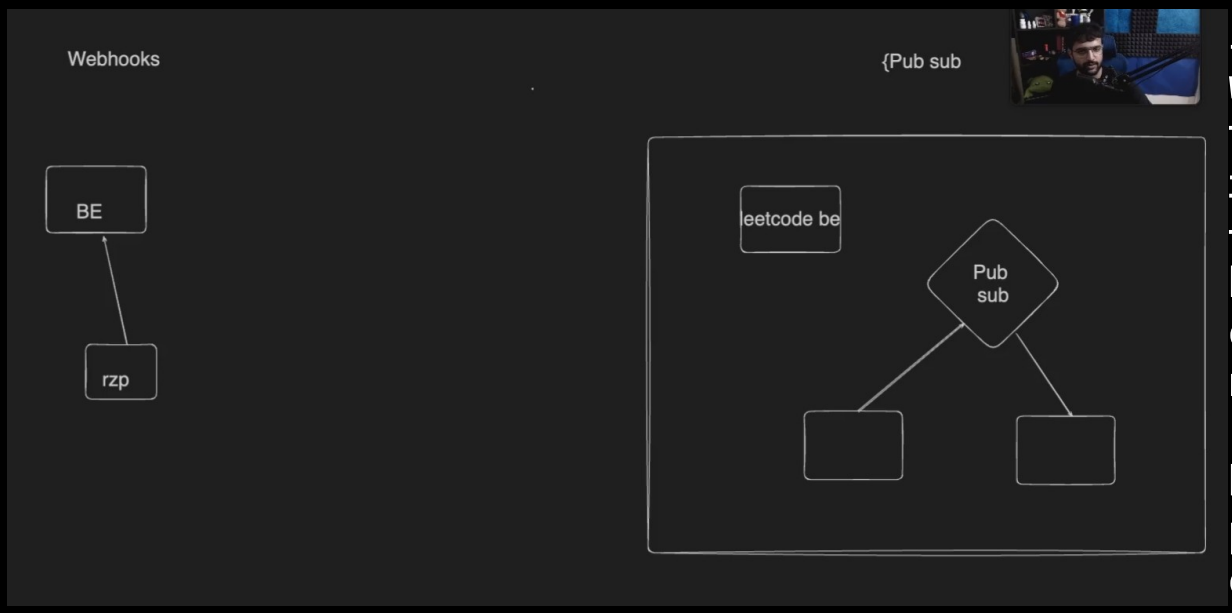
now running postgress locally docker

docker run -p 5432:5432 -e POSTGRES_PASSWORD=mysecretpassword postgres



better schema view you can see

run npx prisma migrate dev
also change the url in .env file
DATABASE_URL="postgresql://
postgres:myscretpassword@localhost:5432/postgres"
name as init



in this the webhooks the http req comes from backend of other user not our tho

but in the pub sub we own the be servers and then we can create the pubsub

added the zaprun in the schema and now adding a queue using redix or kafka

npx prisma generate
pnpm add prisma @types/pg --save-dev
pnpm add @prisma/client @prisma/adapter-pg pg dotenv

whenever changes in prisma use the migrate command and then npx prisma generate also

for now we are declaring that the data we are getting is json only

Prisma studio is shit command not working manually push the commands you want to execute

ATOMICITY - putting data in two be servers
zap run outbox - used for putting the data and keeping in kafka or redis for ready queue

npx prisma format which crates the relations automatically

asynchronous → means like it process take it time to complete the task its not fixed

DAG - directed acyclic graph - similar to linked list? Not sure

react flow - for the zapier interface

cms == customer management system

webhook == its a different system talking to zapier

creating a processor folder and doing
npm init -y
npx tsc --init
uncomment the src and dist folder
create src folder with index.ts

npm install prisma
npx prisma init
copy the prisma file

npm i @prisma/client

npx prisma generate
npx prisma migrate dev
copy the lib folder, package.json, tsconfig file then all
set life jinga lala

using limit 1 in prisma

to run kafka locally , can also use redis queue

if showing error always remove like the second line from the
docker config file from the location

C:\Users\Admin\.docker

docker run -p 9092:9092 apache/kafka:3.7.1

to use kafka we have to create a topic which is like a
creating a queue in the kafka

as we are using docker first we have to get inside the kafka
using this command
docker exec -it 899af514989e /bin/bash
the is we can take from the docker ps
cd /opt/kafka/bin
./kafka-topics.sh --create --topic zap-events --bootstrap-
server localhost:9092 (this will create the kafka topic)

https://kafka.apache.org/quickstart/ (REFERENCE LINK)
inside the processor folder use
npm install kafkajs

```
/*
        [{
            id: "1",
            zapRunId: "2"
        }, {
            id: "1",
            zapRunId: "2"
        }]

*/

producer.send({
    topic: TOPIC_NAME,
    messages: pendingRows.map(r => ({
        value: r.zapRunId
    }))
})
```

this is how the upper two variable is converted into the one value

```
async function main(){
    const producer = kafka.producer();
    await producer.connect();
    while(1){
        const pendingRows = await prisma.zapRunOutbox.findMany({
            where:{},
            take:10
        })

        producer.send({
            topic:TOPIC_NAME,
            messages: pendingRows.map(r =>{
                return{
                    value: r.zapRunId
                }
            })
        })
    }
}
```

the pendingrows get the data from the db about the zaprunoutbox

then push it to kafka

and then later we have to delete also

trying to add the data
docker exec -it 899af514989e /bin/bash
cd /opt/kafka/bin

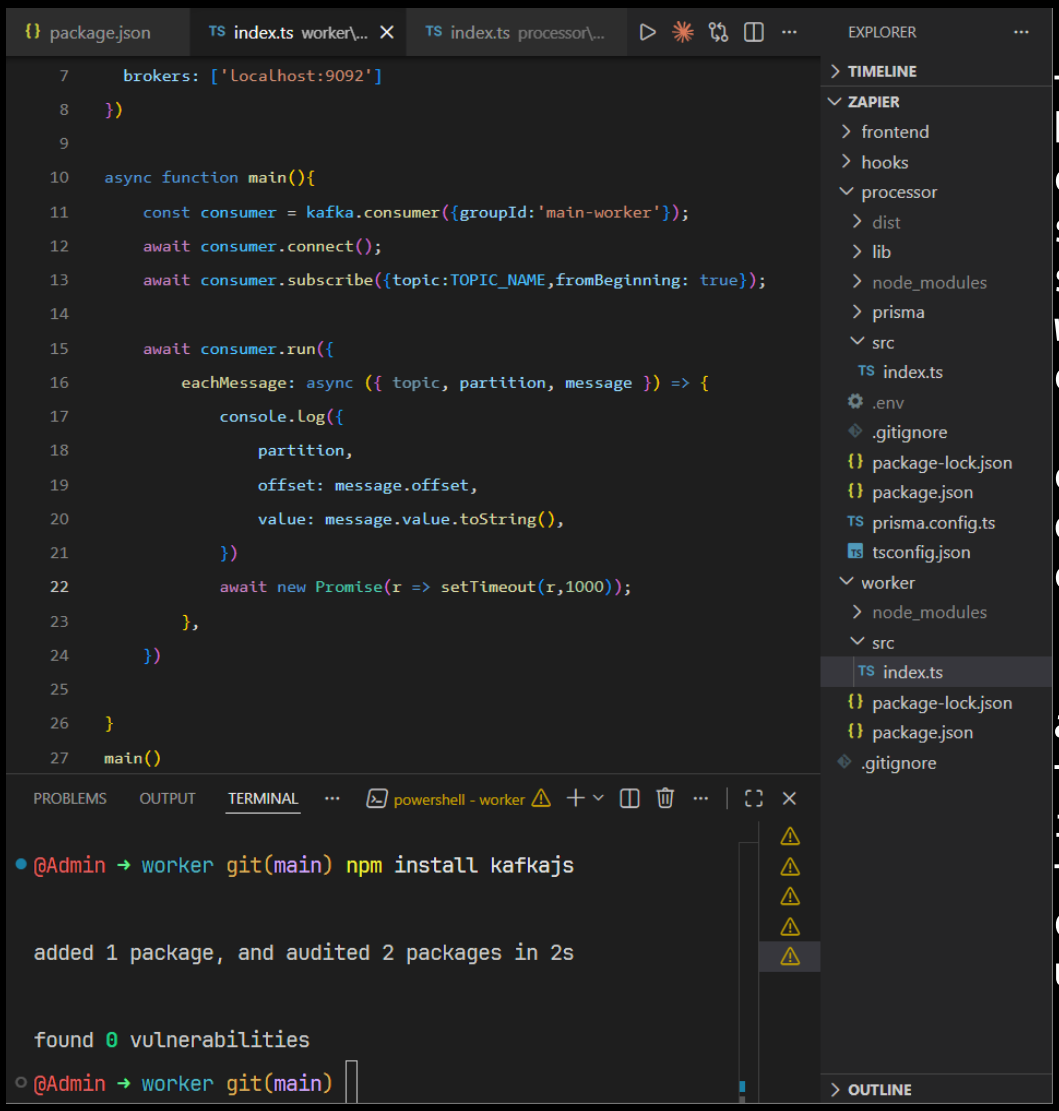
```
./kafka-console-consumer.sh --topic zap-events  --from-beginning --bootstrap-server localhost:9092
npx prisma studio
```

the working here is like whenever the data is pushed to the kafka then the data is deleted from the db

every zap should be sorted so adding to schema

now this is made now we have to make a kafka processor

worker is the processor which gets called to send emails to users something like that



first intialize a kafka consumer connects to it subscribe on a specific topic whatever events are coming send to this

consumer.run is every time message comes log it

await because as the email sending is a expensive task to reduce the time of exec we are using the promise

now there is a new issue like you pulled the data from the kafka queue but does not process it to fix that now
like in this our worker dies

we have to do like until the worker sends back the work is done don't mark it as done

by default the there is a autocommit in kafka its a process of acknowledging and telling that move on to the next so we need to disable that

docker run -p 9092:9092 apache/kafka:3.7.1 (this is to use kafka)
to use kafka we use
docker exec -it b564e34c485b /bin/bash

adding data in kafka queue
cd /opt/kafka/bin/
./kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
in this we have to add the data like random words its like something which creates the queue of data

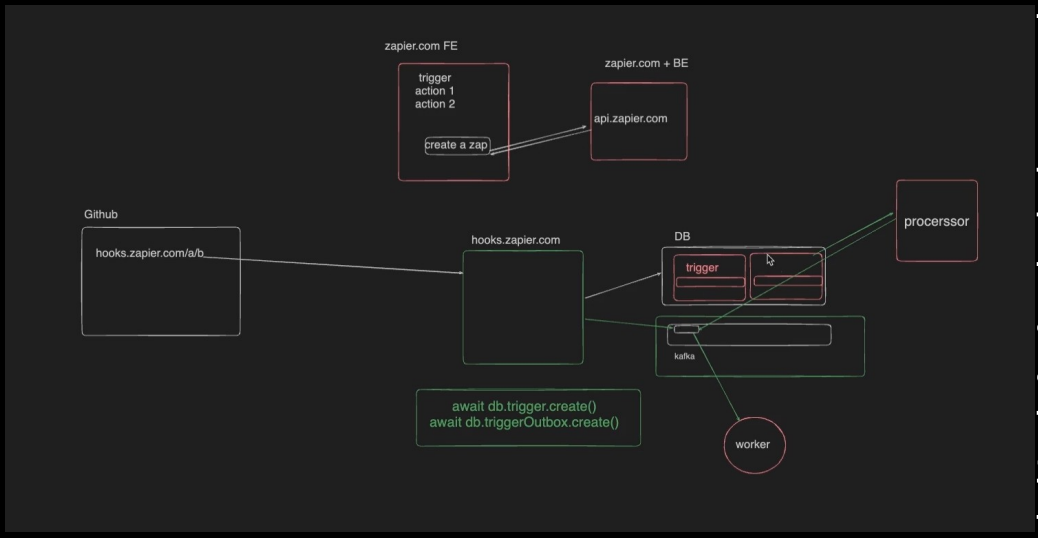
```
23          await new Promise(r => setTimeout(r,5000));
24          console.log("processing done");
25          await consumer.commitOffsets([{
26              topic:TOPIC_NAME,
27              partition:partition,
28              offset: String(Number(message.offset) + 1)
29          }])
30
31      },
32  })
```

this parts help in like if the data from the queue is picked but somehow the server crashed in between than the worker will start picking the data from the crashed one only it will not leave that

like if the data crashed during offset number 2 that means from next start it will work from offset 2 only

```
upProtocol":"RoundRobinAssigner","duration":23288}
{ partition: 0, offset: '12', value: 'asdf' }
processing done
{ partition: 0, offset: '13', value: 'sdf' }
@Admin → worker git(main) ☹ npm run start

> worker@1.0.0 start
> tsc && node dist/src/index.js

{"level":"INFO","timestamp":"2026-04-01T06:57:40.801Z","lo
gger":"kafkajs","message":"[Consumer] Starting","groupId":
"main-worker"}
{"level":"INFO","timestamp":"2026-04-01T06:58:07.508Z","lo
gger":"kafkajs","message":"[ConsumerGroup] Consumer has jo
ined the group","groupId":"main-worker","memberId":"outbox
-processor-5a5803e2-ff42-4093-86f8-3b9af8a03ead","leaderId
":"outbox-processor-5a5803e2-ff42-4093-86f8-3b9af8a03ead",
"isLeader":true,"memberAssignment":{"zap-events":[0]},"gro
upProtocol":"RoundRobinAssigner","duration":26704}
{ partition: 0, offset: '13', value: 'sdf' }
```

see in this example same thing is happening we crashed the offset 13 value and it started from that



the use of this zapier part is just to pick data from the db and then putting to the kafka queue and nothing else and once they are transffered the request is fullfilled

DIFFERENCE BETWEEN KAFKA AND REDIS
kafka lets you creates bunch of machines
scale horizontally, add bunch of brokers to the clusters
like if one of the brokers go down you are sorted
gives better fault tolerance
Kafka also has partitions which lets to scale horizontally
its also has consumer groups

MAIN BENEFIT OF USING KAFKA IN THIS IS LIKE
When like a person creates a zap which does like github comment to send bounty which like takes the solana address from him and then send them notification this must be done in sequence which kafka handles by creating partition for each sequence rather than assinging them different workers

the idea is single zap execution should never go to sepearte partition for that reason for each whole sequence a single worker only pull the work

LONG FUCKING REASON

kafka also maintains a history like for the all the offsets processed after completion also it can play from starting again, lets you replay messages

to check for the port if something is running
netstat -ano | findstr :3000

and then take the id and kill it using this
taskkill /PID 1528 /F

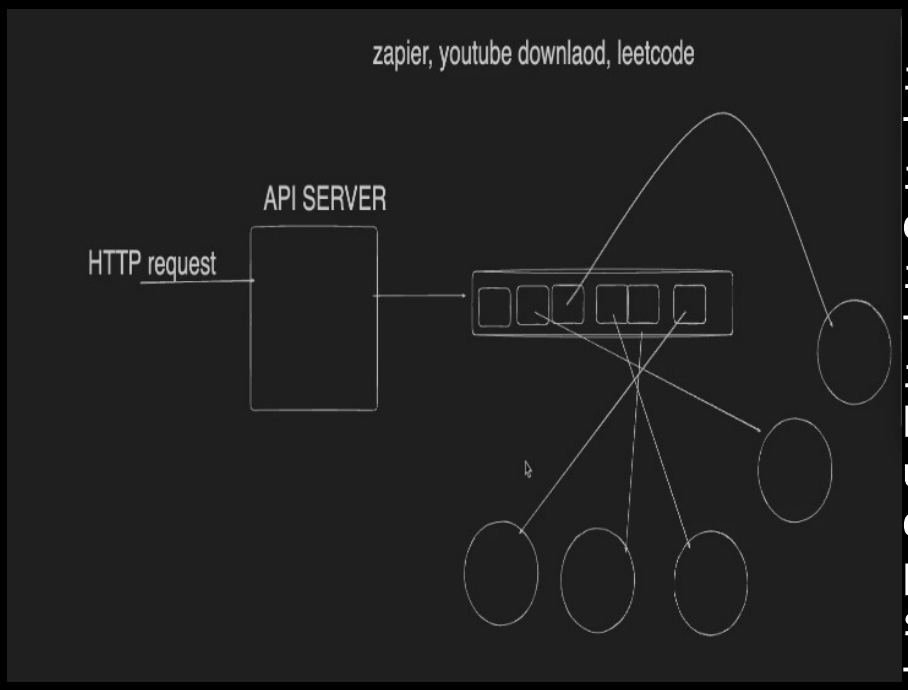
now what we did is ran both docker files postgress and kafka and then ran all three folders

and then hitting the url
http://localhost:3000/hooks/catch/1/a0154130-6c87-404c-b7b7-137c57b33767

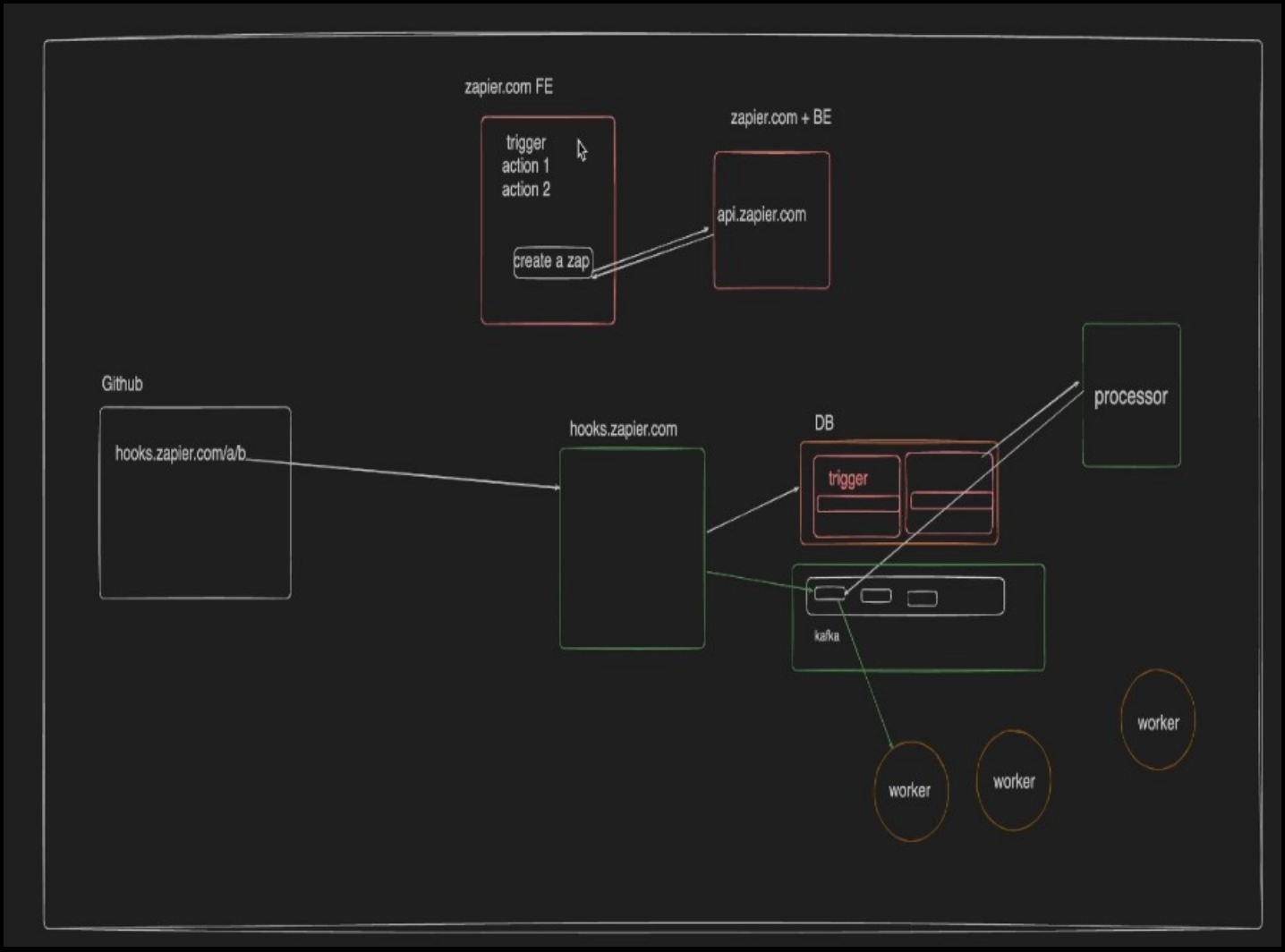
from this when we hit the data is received in the kafka queue and then pushed to the worker offset

to add support to multiple other services we have to add the logic in the processor file

KAFKA AND WORKER RECAP



in this its like we get the data and we process it and add it to the queue and whenver there is a overload of usage the workers can be increased as it can be horizontally scaled using the kafka so its done to offload the pressure from the api server and prevent it from chocking



The below part is done the upper part is left to create
create a primary -backend folder
npx tsc --init
npm init -y

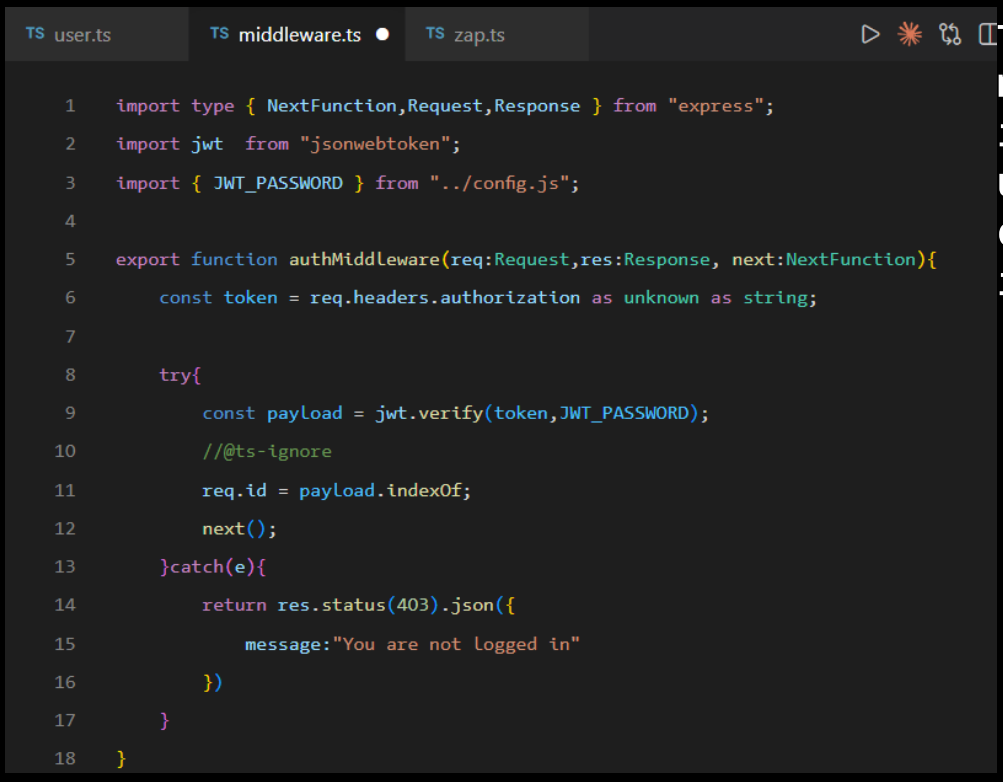
yarn add express @types/express
npm install cors

npm install @types/cors

npx prisma init

copying the schema from the hooks and also isntalling the zod for type checking

```
npx prisma generate
npm install @prisma/client @prisma/adapter-pg pg dotenv
npm install jsonwebtoken
npm install @types/jsonwebtoken
npm i --save-dev @types/node

export const JWT_PASSWORD = process.env.JWT_PASSWORD ||
"123random"

what this above thing means?
It will be either a env var called JWT_PASSWORD and it that
dne then use the 123random

//@ts-ignore USED TO REMOVE THE HEAVY CHECKS

req.headers.authorization is what the user sends to the
middleware
```



this is how the middleware looks and it just verifs if the user is that user only which is present in the db

```
docker run -p 5432:5432 -e
POSTGRES_PASSWORD=mysecretpassword postgres
npx prisma migrate dev
npx prisma studio == to view the user
```

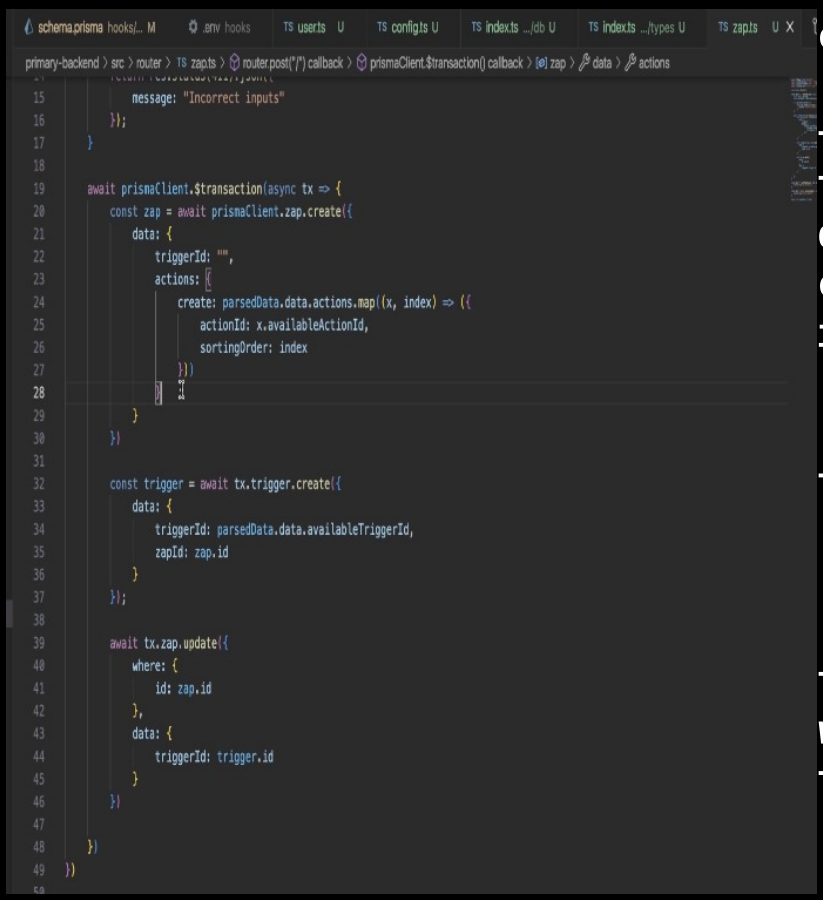
What all routes are working right now?

On post request
http://localhost:3000/api/v1/user/signup
this route works and data should be provided in this format
{
    "username":"pallab11@gmail.com",
    "password": "12312311",
    “name”:”Pallab”
}

http://localhost:3000/api/v1/user/signin
this url returns the token for the user and remove the name from this

http://localhost:3000/api/v1/user/

for this the toke is required in the headers in the authorization to retrieve the data



creation of zap-events

firstly bound inside the transaction and also the data which is firstly create the zap with a random trigger id

then create the trigger

then update the trigger with the new zapid and trigger id

as this is happening inside a transaction is any steps fails whole code stops

npx prisma migrate dev (TO GET THE SCHEMA IN THE POSTGRES)
npx prisma generate (CREATE THE FILE)
npm run dev
npx tsx prisma/seed.ts (SEED THE DUMMY DATA)
npx prisma studio (VISUALIZE)

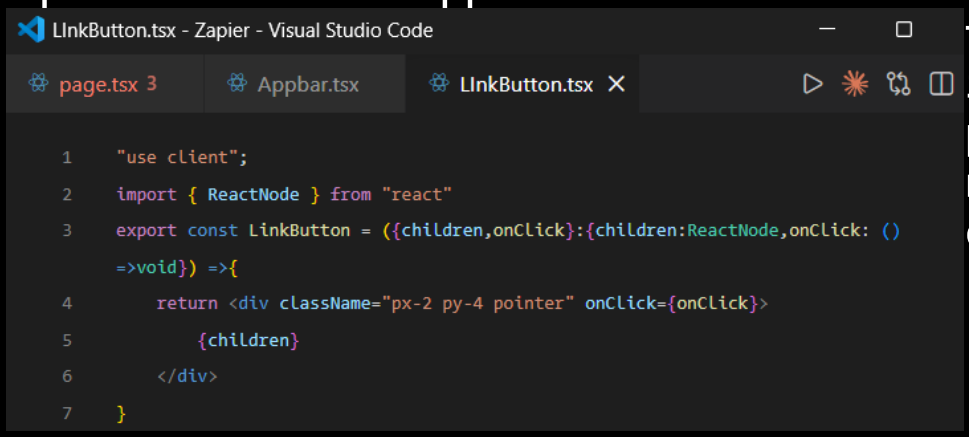
before running the the zap first create the user using this
1. Sign up:
POST http://localhost:3000/api/v1/user/signup
```
{
    "username": "test@test.com",
    "password": "password123",
    "name": "Test User"
}
```
2. Sign in (to get a valid token):

POST http://localhost:3000/api/v1/user/signin
```
{
    "username": "test@test.com",
    "password": "password123"
}
```
3. Then use the returned token in the Authorization header for the zap creation request.

Your old token references a user that doesn't exist anymore (likely the DB was recreated during migrations).

THEN POST REQUEST ON
http://localhost:3000/api/v1/zap
WITH THE TOKEN IN THE AUTH FROM THE ABOVE WE GET
```
{
    "availableTriggerId": "webhook",
    "triggerMetadata":{},
    "actions":[{
        "availableActionId":"email",
        "actionMetadata":{}
    },{
        "availableActionId":"sol",
        "actionMetadata":{}
    }]
}
```
AND THEN THIS IN THE BODY AS THE DATA
THEN WE GET A ZAP ID

CREATING THE FRONTEND
using next js
npx create-next-app



this creates children in which we can provide the button name and also the onClick function

use router is only worked with use client
pr == padding towards right

font weight == for overall boldness of the text
max width for the overall width from each side
pt-4 , pl,pr can be padding from top , left , right and bottom

flex justify center for the two buttons too be sepearated with a distance

using heroicons

Adding all the data and e.target.value to all the buttons and creating the login and signup pages properly and in the login page removed the name thingy

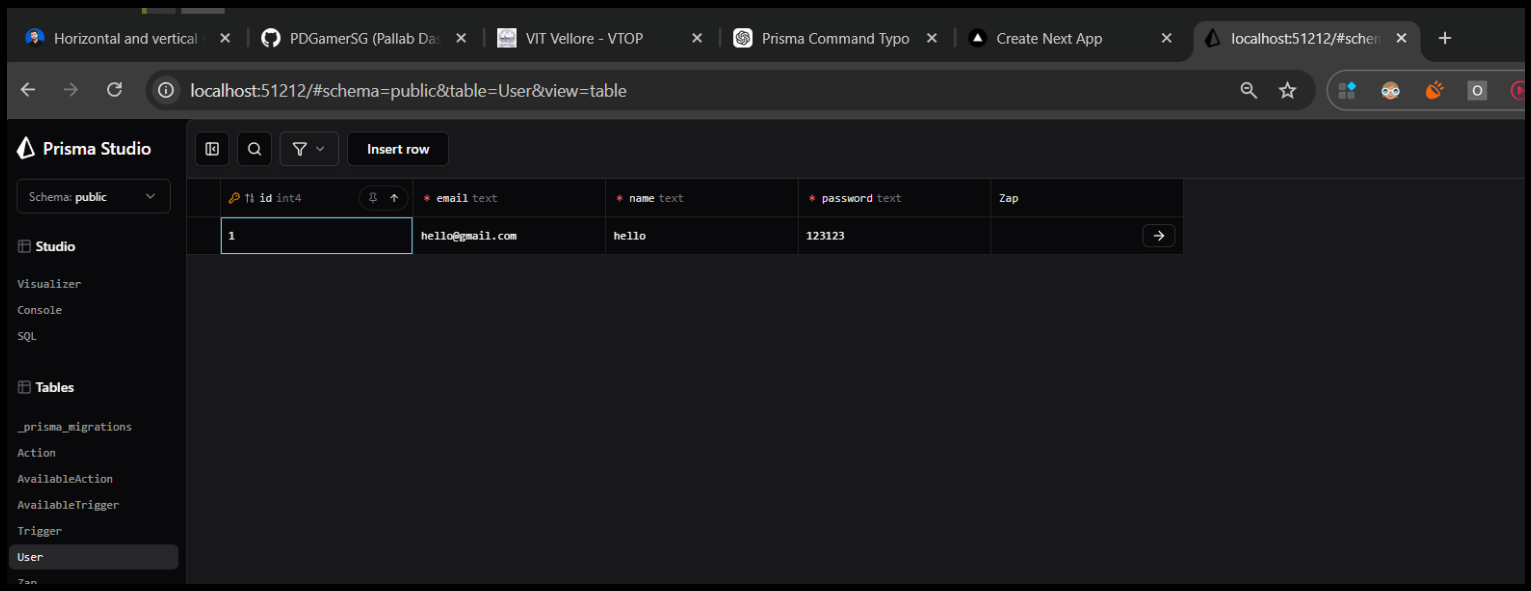
now adding axios request to fetch the data from the frontend to push to backend

npm install axios then

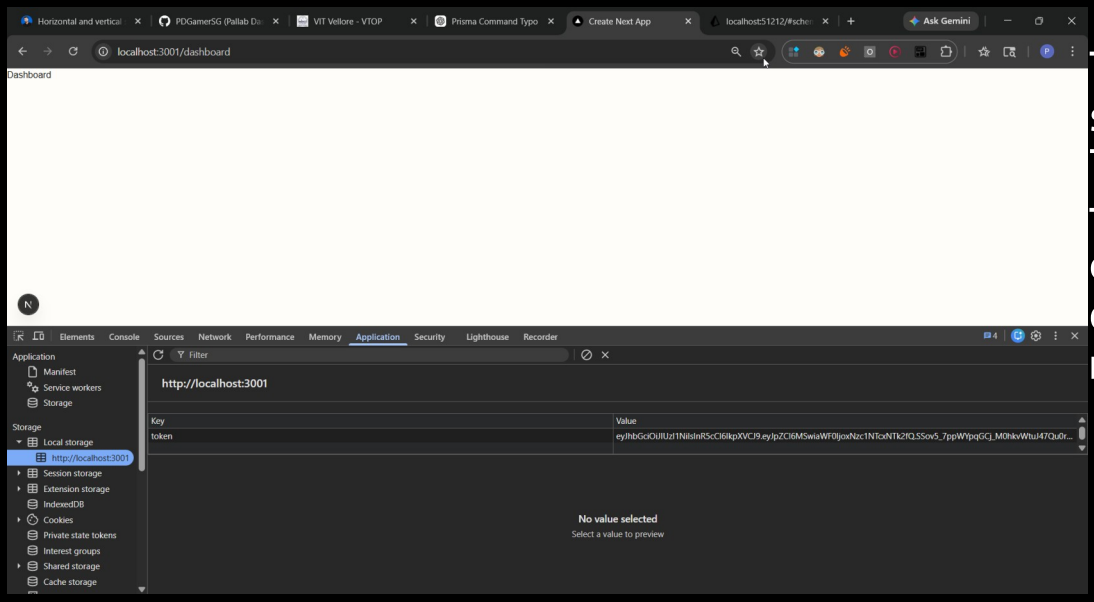
keep in mind that the use Router which is used should come from next/navigation

after a successful signup just redirect to login page and after reaching the login page first store the token in the localstorage and then use it

CHECKING ALL THE ENDPOINTS
trying to check all the endpoints in this to run the backend you have to run the backend first which will run on the 3000 port with the docker also running
then running the frontend



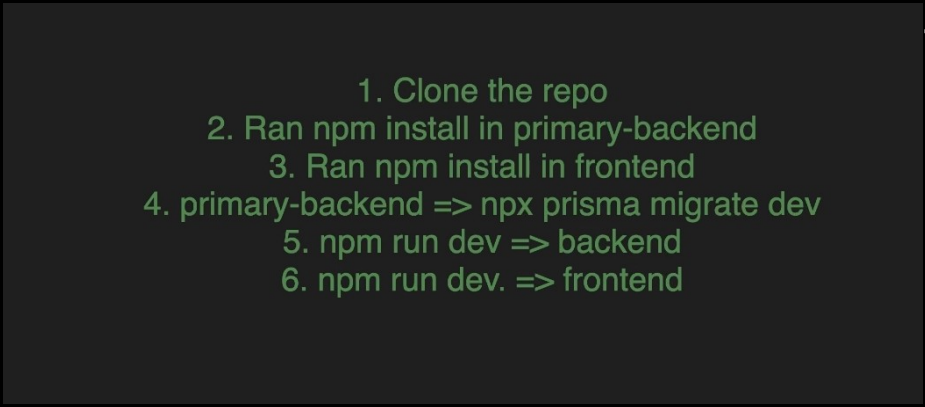
now when you will register in the user It will show in the db using the npx prisma studio
NOW LOGIN AND THE SIGN IN PAGE IS ALSO WORKING



Token is also stored in the local storage for the website to check it is the original user or not

DASHBOARD
completed the dashboard nothing fancy and in that we also created like it was easy nothing fancy and also in this now we are creating the zap/create page

using the react flow



these are the steps to start the project

created the available triggers the actions and triggers in the index.ts page
with each trigger there is a image in that we can see which thing is used

added available action and triggers image url and saved them

http://localhost:3000/api/v1/trigger/available

this in the postman returns all the triggers
http://localhost:3000/api/v1/action/available

this returns the actions

now in the zap/create/page.tsx creating the modal function to help the user choose the trigger

now we are using modal for this part we can also use shadcn to work with also
copied the modal code and using that and printing it whenever the user selects that

to check the ui if working or not try to put alert(“hi”) like this to check the errors

now creating the custom hook for the useAvailableActionsAndTriggers() which fetches all the available action and triggers which we can show in the card

getting back the data for this from the backend from the urls

now created the flow also and then the flow is shown in the dashboard also

now creating it will ask for metadata also like solana address to which to we have to send the data

whenever changes in the db need to migrate every time also