Practical work to start

Installations needed

<u>Step by step to create a New Server with installations first</u>

Update your system
Command: sudo apt update && sudo apt upgrade -y

Install Docker:
Command: sudo apt install docker.io -y

Enable & start Docker
Command: sudo systemctl enable docker
Command: sudo systemctl start docker
Command: sudo systemctl status docker

Give Docker Permissions to current user
Command: sudo usermod -aG docker ubuntu

Command: logout

Again login into the server.

Now install Docker Compose

Command: sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Command: sudo chmod +x /usr/local/bin/docker-compose
Command: docker-compose --version

Install Postgresql

Command: sudo apt install postgresql postgresql-contrib -y

Start and Enable PostgreSQL
Command: sudo systemctl start postgresql
Command: sudo systemctl enable postgresql
Command: sudo systemctl status postgresql

Create PostgreSQL user

Command: sudo -u postgres psql

Inside the prompt, run:
Command:
CREATE USER postgres WITH PASSWORD 'admin123';
ALTER USER postgres WITH SUPERUSER;
\q

Note: above create user means creating postgres as user and giving password as admin123

so,
Now test the PostgreSQL Login

Command: psql -U postgres -h localhost

Note: if its ask password then admin123

If you get a connection error
Command: sudo nano /etc/postgresql/*/main/pg_hba.conf

Change this line : local all postgres peer

TO

To this : local all postgres md5

Then restart the postgreSQL
Command: sudo systemctl restart postgresql

Now install Node.js and npm

Command: sudo apt install nodejs npm -y
Command: node -v
Command: npm -v

Command: curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

Install node.js Dependencies

Command: npm install express cors multer dotenv path fs morgan helmet express-rate-limit
To check all the services status
Command: docker --version
Command: docker-compose --version
Command: systemctl status docker
Command: systemctl status postgresql
Command: psql -U postgres -h localhost
Command: node -v
Command: npm -v

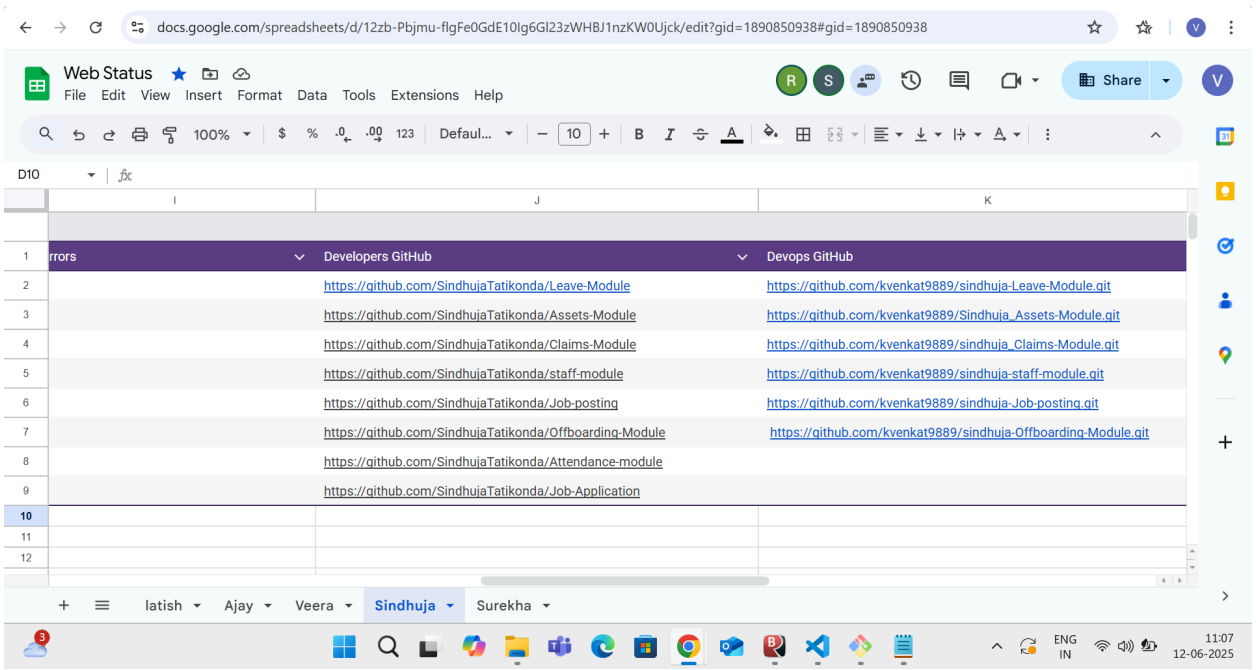
Developers work

Once the Developer writes the code he/she will share the link to one updating page which have created.

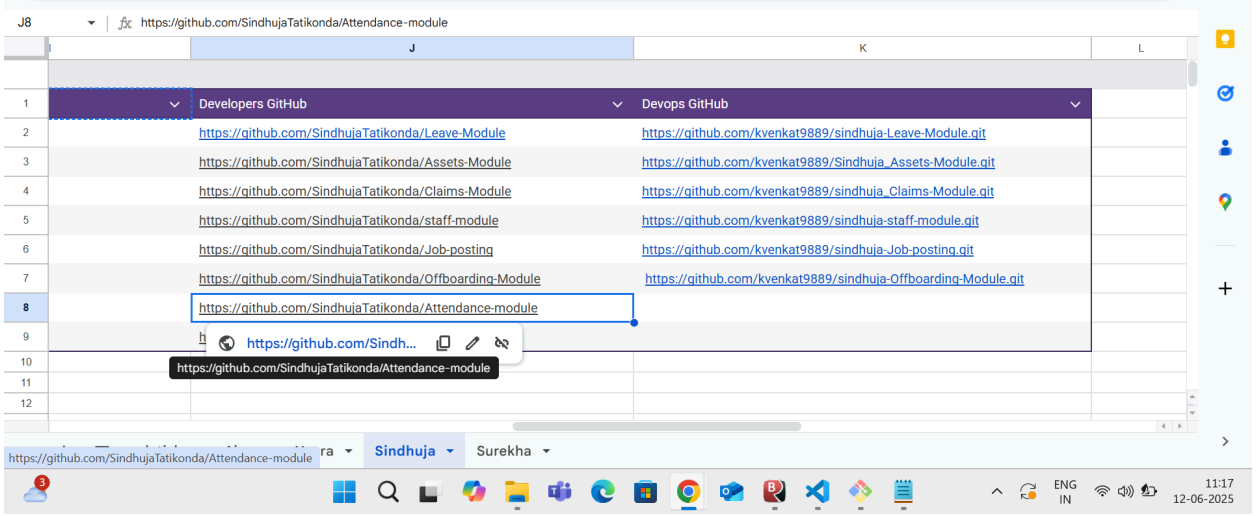
For Example: Created this sheet to keep the updated new link codes to take easily from below screen shot (sheet).

1. In this sheet you can see,

WEB Pages	Status	Developer	URL's	URL's created AWS Devops Engineer	Post Date	Completed Date	Reviewed by Tester	Errors

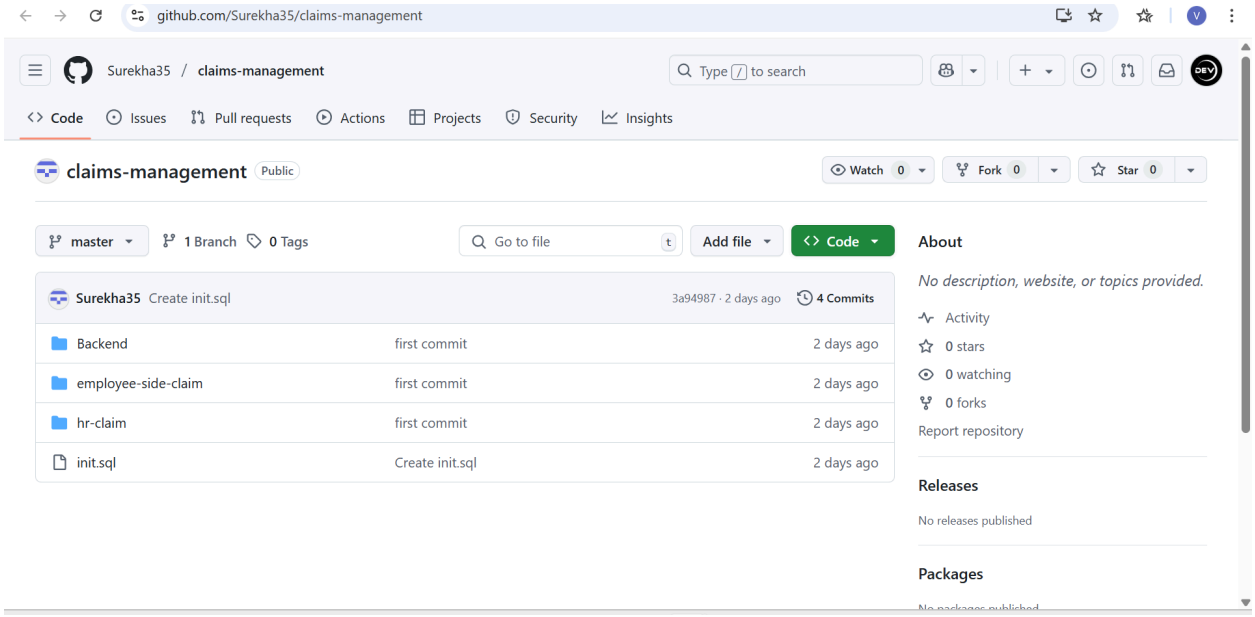


Once you can on the link it will open the Developer Github link

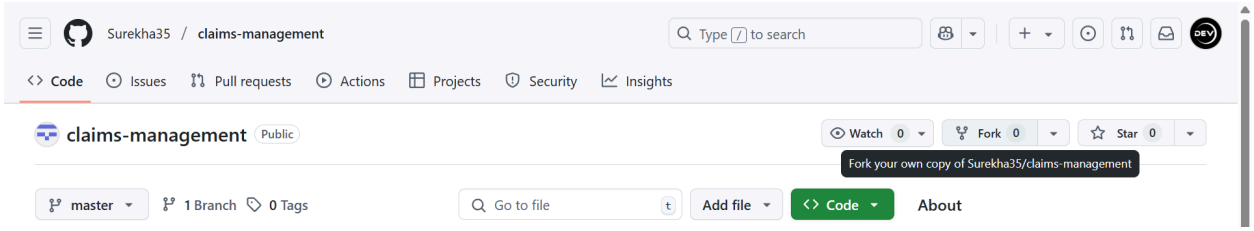


	Developers GitHub	Devops GitHub
2	https://github.com/SindhujaTatikonda/Leave-Module	https://github.com/kvenkat9889/sindhuja-Leave-Module.git
3	https://github.com/SindhujaTatikonda/Assets-Module	https://github.com/kvenkat9889/Sindhuja_Assets-Module.git
4	https://github.com/SindhujaTatikonda/Claims-Module	https://github.com/kvenkat9889/sindhuja_Claims-Module.git
5	https://github.com/SindhujaTatikonda/staff-module	https://github.com/kvenkat9889/sindhuja-staff-module.git
6	https://github.com/SindhujaTatikonda/Job-posting	https://github.com/kvenkat9889/sindhuja-Job-posting.git
7	https://github.com/SindhujaTatikonda/Offboarding-Module	https://github.com/kvenkat9889/sindhuja-Offboarding-Module.git
8	https://github.com/SindhujaTatikonda/Attendance-module	

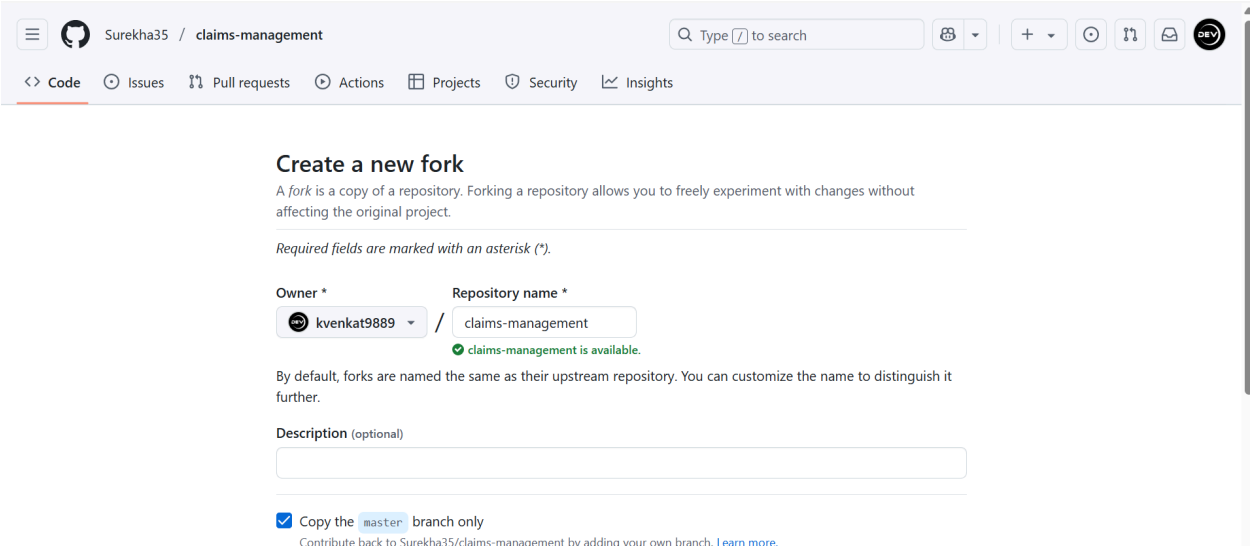
Below screen shot is Developers web page link



First you need to FORK the page, means you need to create a Repository to get that code copy to your github link.



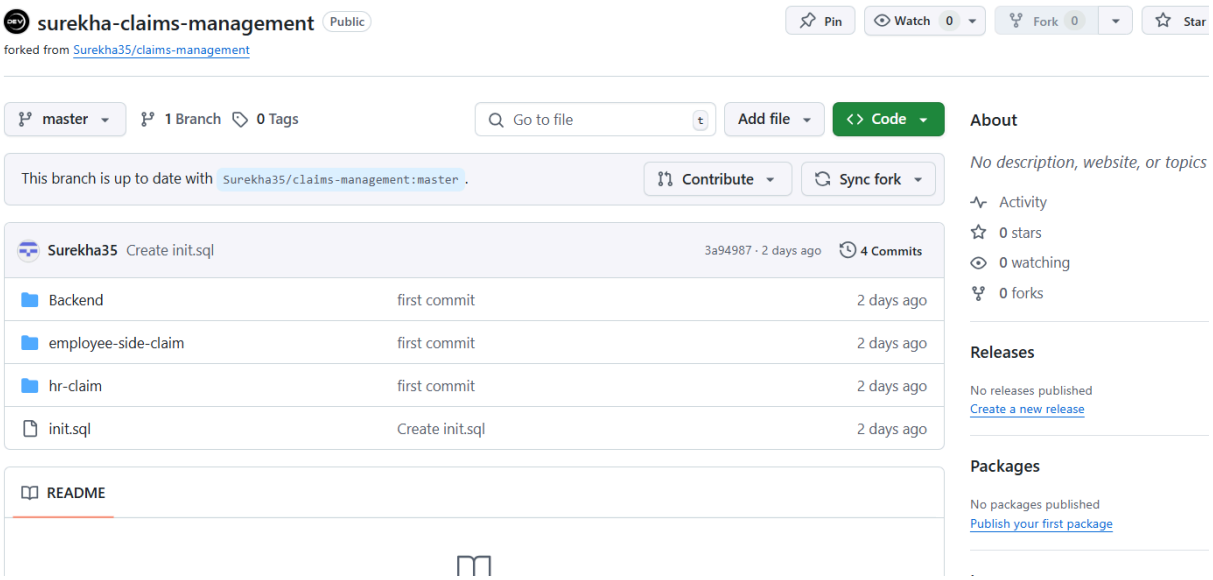
Once you click on the fork, it will redirect the page to create a Repository name



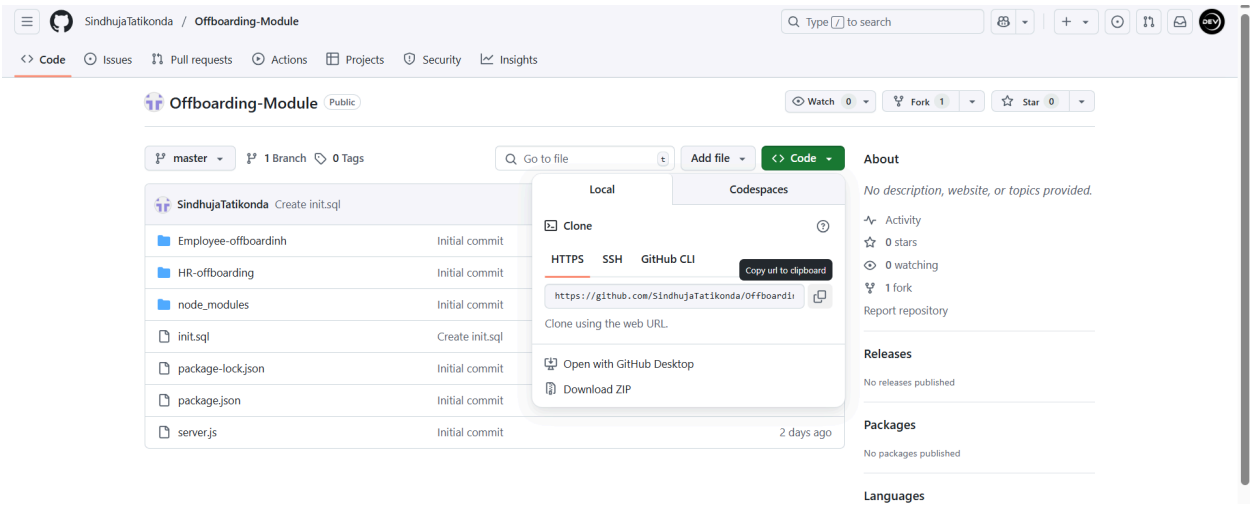
Then change the name as per you need to identify.
For example: I created a repository name as surekha-claims-management to identify easily because of its surekha code.

Then After click on **Create Fork**

Once you create a Fork, it will open your page with the name which you have created with.



Now, click on the **Code** copy the link, screen shot you can see below



Open your visual studio code
Click on File > New window > **Clone Git repository >** Paste the link above and click on **Enter**
And save the file where you can save
Click on **Open**
Now you can see the code in visual studio code

Note : files should be like this, Developers should send in this format you can see below

Backend (Folder) inside backend below files should there
. package.json
. package-lock.json
. server.js
. node-modules

Employee or frontend or any name they keep, in this folder they should be index.html file
. index.html (file)

Hr or any other name (in this also index.html should be there).
. index.html

Init.sql (file) its database query to create a table in database

Note: init.sql very important to have this file to create table in database

If they give you 3 pages code also
For example above you having employee side code and Hr side code know

Offerletter (folder) in this folder also index.html should be there

. index.html

How many pages are there that many folders need to have index.html file.

In Visual code,
Now we need to make some as per our need
First we need to ask the Developer which is the number you created with backend port

For example: now developer created a backend port with **3000**
This is how it look likes

```
const express = require('express');
const { Pool } = require('pg');
const cors = require('cors');
const app = express();
const port = 3000;
```

Const port = 3000 (its a backend port they have given)

We need to change as per our need,
For example : if its your starting page to start with for you can go with 3001 port
So, change to 3000 to 3001 port

If not already you have given 3001 port to other page means then give 3002 port or 3003 anything you want give,

Note: Just remember that, once given port 3001 to one page means the same 3001 port will not work to other page servers.

Now lets change the **Database credentials** server.js **file**

```
// PostgreSQL connection configuration
const pool = new Pool({
    user: 'postgres', // Replace with your PostgreSQL username
    host: 'localhost',
    database: 'employee_claims_db',
    password: 'root', // Replace with your PostgreSQL password
    port: 5432,
});
```

The above screenshot shows Developer credentials

So, we need to change the credentials as per our database credentials
You can see below
I changed only
Host: postgres
Password: admin123 (my database password)

note : here you can see 3801 port know, its my port number i have given and its your choice to port number.

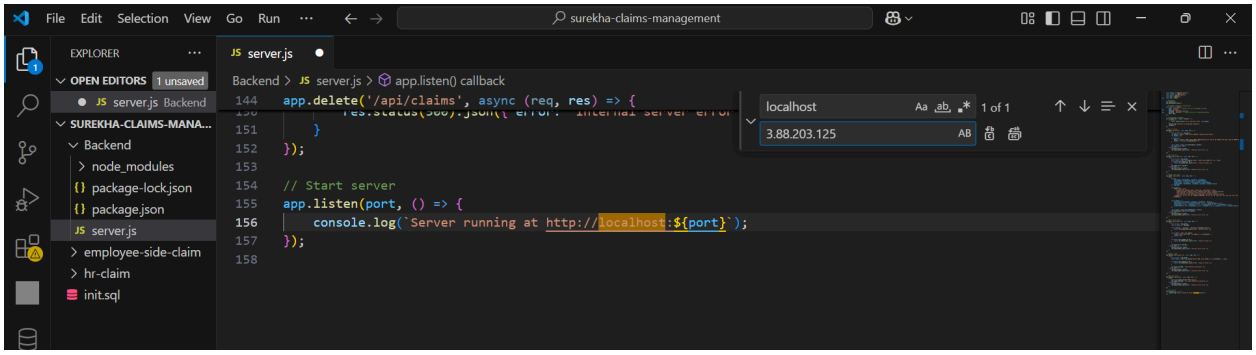
```
const express = require('express');
const { Pool } = require('pg');
const cors = require('cors');
const app = express();
const port = 3801;

// Middleware
app.use(cors());
app.use(express.json());

// PostgreSQL connection configuration
const pool = new Pool({
    user: 'postgres', // Replace with your PostgreSQL username
    host: 'postgres',
    database: 'employee_claims_db',
    password: 'admin123', // Replace with your PostgreSQL password
    port: 5432,
});
```

Now again give cntrl+f and in search bar give localhost it will show you where the localhost is.

Now in the place of **local host** you need to replace with your **public ip address**
You can see below



Changed the localhost to ip address

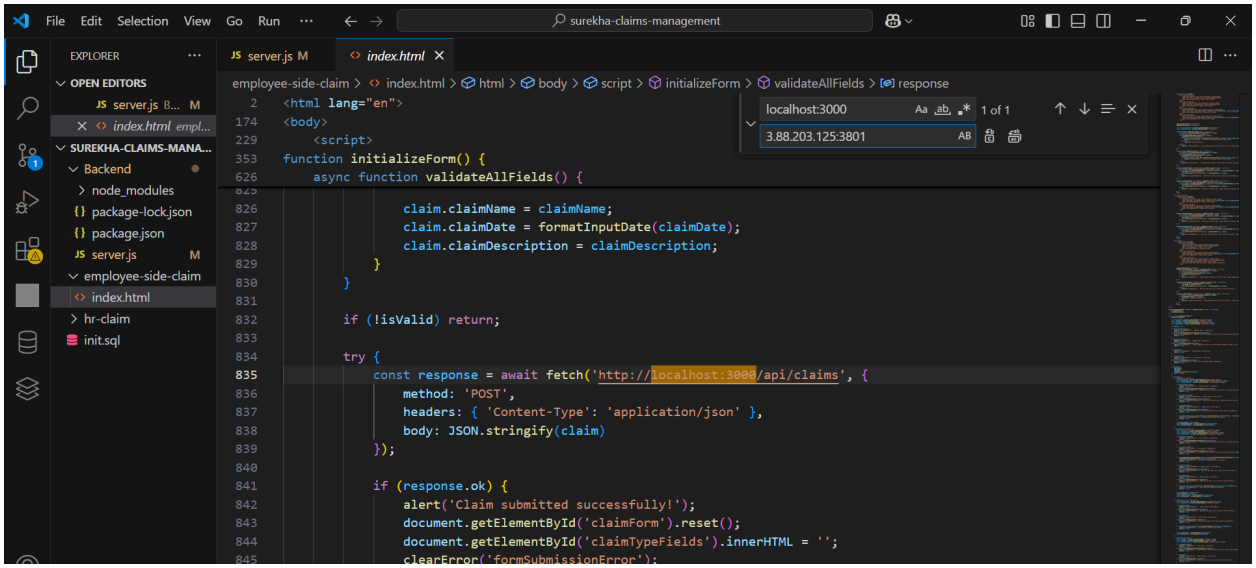
Once completed with this part

Cntrl + s =(save)

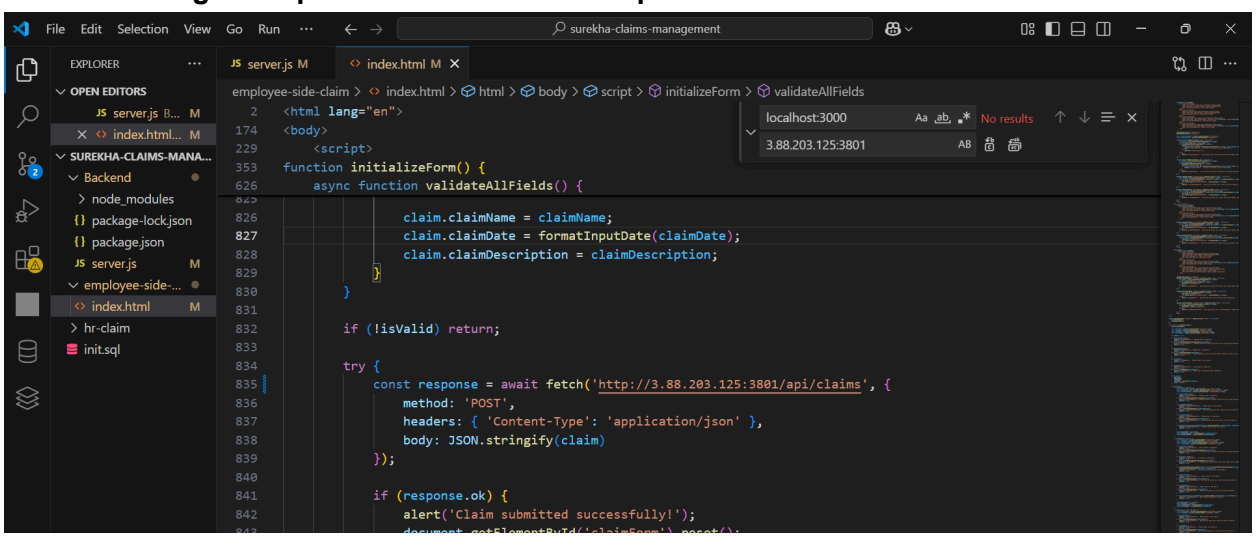
Now open the
employee side folder and click on index.html

In index.html code need to make some changes

Note: only we need to change localhost:3000 to our public ip address (3.88.203.125:3801) with my backend port number 3801. Click on Enter



Once its changed to ip address with backend port

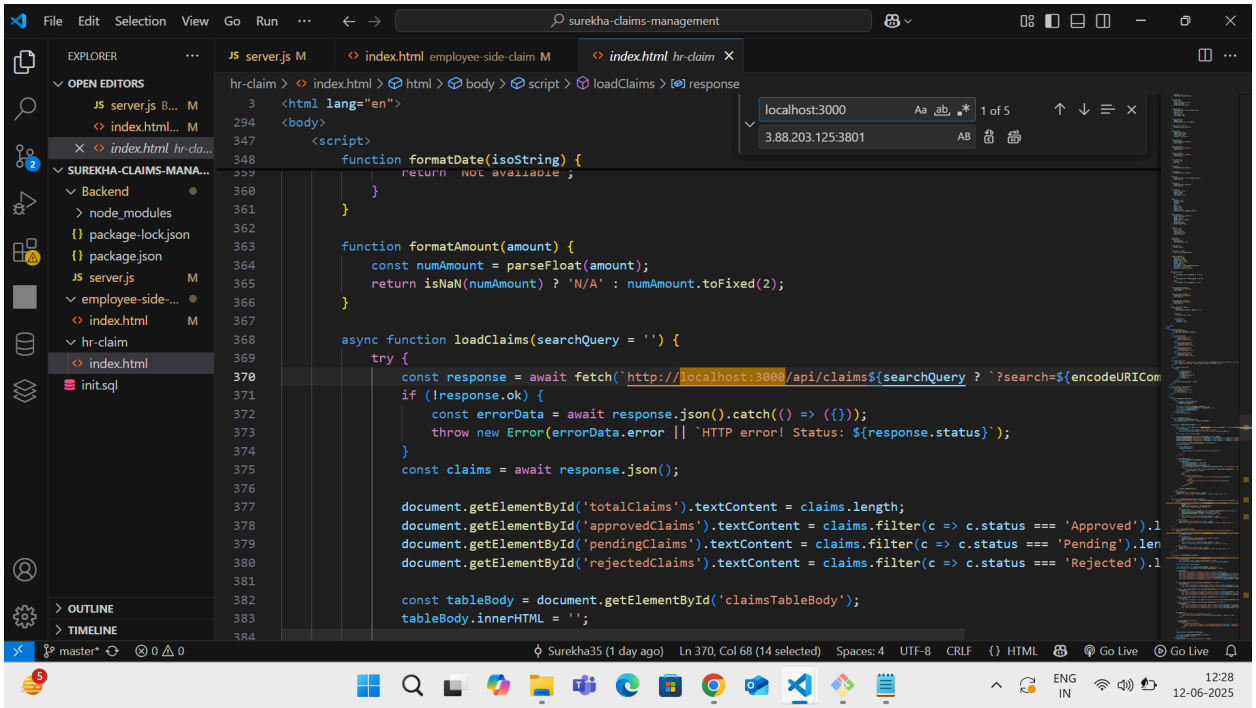


Now **cntrl + s** save the changes

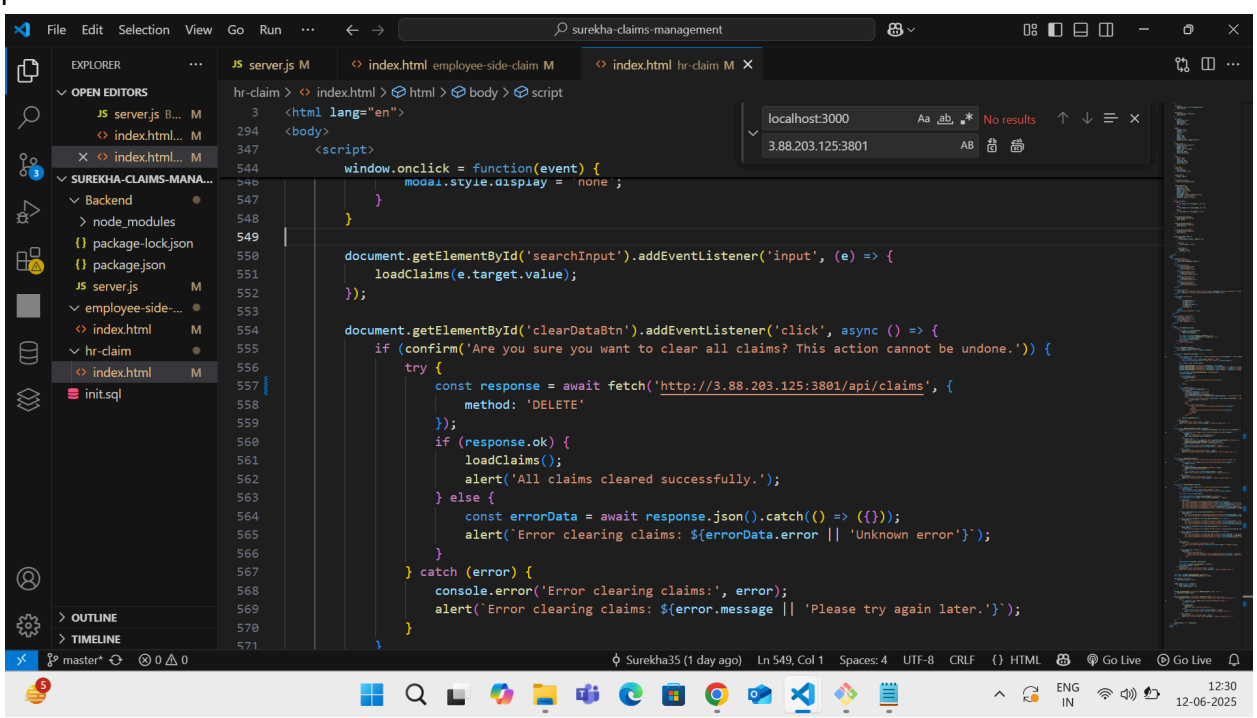
Same thing need to do in **HR** side **index.html files** lo

Open Hr side index.html

Replace localhost:3000 with ipaddress with backend port number



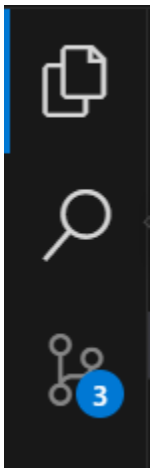
Note: above you can see 1 of 5 on right top
It means localhost:3000 are totally 5 so in that places you need to change with ipaddress and port number.

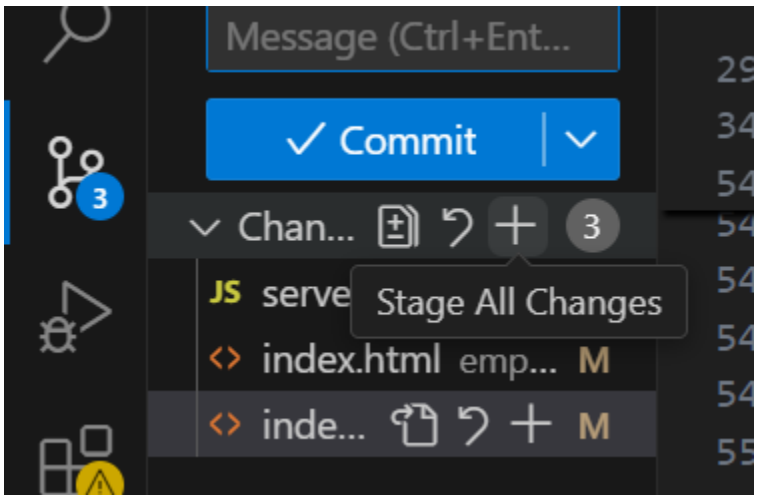


Once you done with that
Cntrl + s = save changes

Now you can see the on left **source control** has some changes made in code, so we need to push the changes to our github link where we created the respiratory name as **surekha-claims-management** .

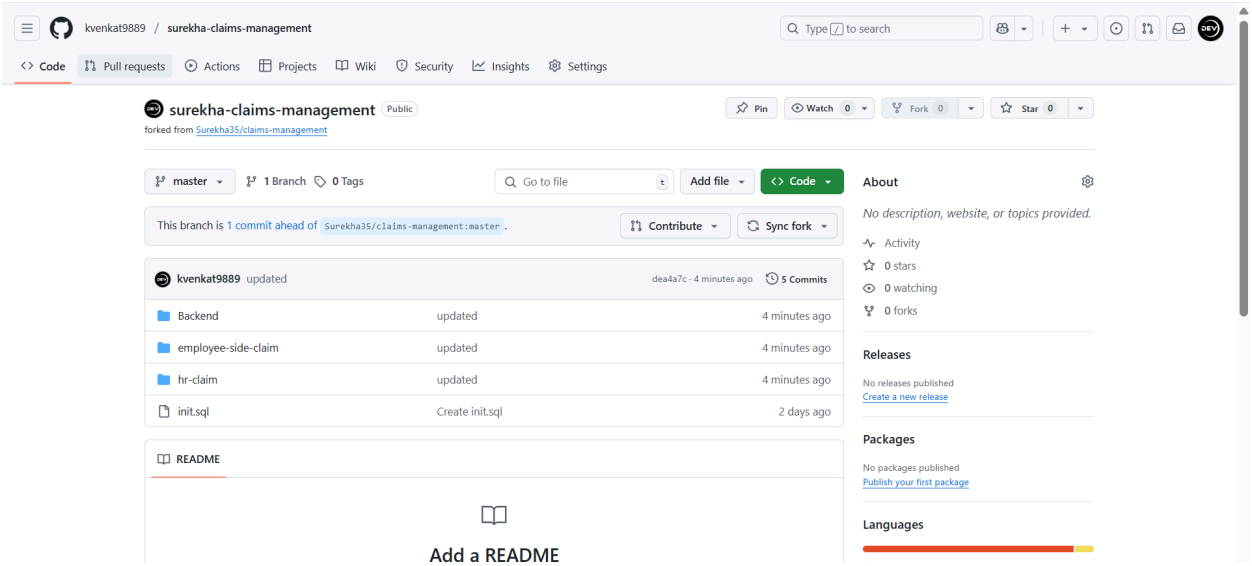
Here source control have **3 changes**, click on **source control symbol**





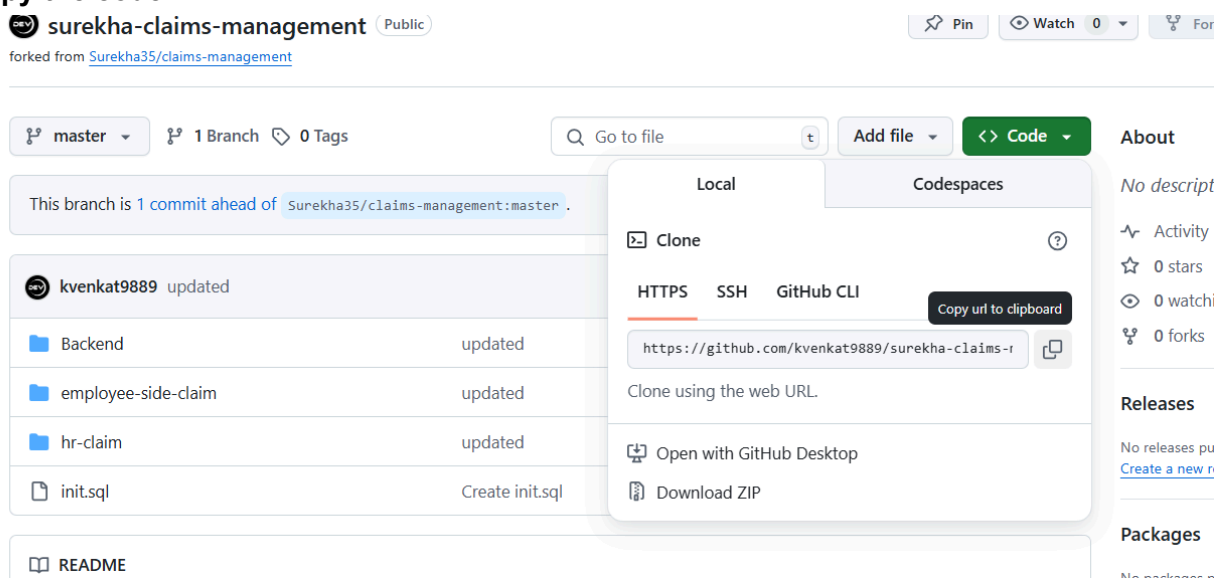
Here you can see changes beside **+** click on that **plus(+)** to **stage all changes**
Then give **message** to commit
Click on **commit**
Click on **sync changes** then click on **ok**

Go Back and see in the github, you can see the changes you pushed 'now' or minutes ago like that.



Now start the Process to work and create a the URLs for this pages

Copy the code link



Open gitbash

1. **Sign into the server with ipaddress**

Command : Git clone https://github.com/kvenkat9889/surekha-claims-management.git
Command : ls

```
Last login: Wed Jun 11 04:40:35 2025 from 49.204.14.110
ubuntu@ip-172-31-84-115:~$ git clone https://github.com/kvenkat9889/surekha-claims-management.git
Cloning into 'surekha-claims-management'...
remote: Enumerating objects: 2255, done.
remote: Counting objects: 100% (2255/2255), done.
remote: Compressing objects: 100% (1838/1838), done.
remote: Total 2255 (delta 273), reused 2250 (delta 271), pack-reused 0 (from 0)
Receiving objects: 100% (2255/2255), 6.27 MiB | 21.82 MiB/s, done.
Resolving deltas: 100% (273/273), done.
ubuntu@ip-172-31-84-115:~$ ls
Ajay  latish  lsurekha_eave-management  sindhuja  surekha-claims-management  surekha-pages  surekha_asset_requests  surekha_offboarding  veera
ubuntu@ip-172-31-84-115:~$ cd surekha-claims-management
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ ls
Backend  employee-side-claim  hr-claim  init.sql
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ cd Backend
ubuntu@ip-172-31-84-115:~/surekha-claims-management/Backend$ ls
node_modules  package-lock.json  package.json  server.js
ubuntu@ip-172-31-84-115:~/surekha-claims-management/Backend$ cat server.js
```

Command : cd surekha-claims-management
Command: ls

Note : here we need to create a docker-compose.yml files to create the containers with port numbers

To run the multiple containers using a single docker-compose file

Command: vi docker-compose.yml
Inside docker-compose.yml file you need to write the code

Note : we need to create a docker-compose.yml file inside file, to run the containers we need to give employee-side-claim port 8102 and hr-calim port 8103 and we need to give database postgres port 5801:5432 and we need to give backend port 3801:3801 and also we need to create a Dockerfile inside Backend folder and create Dockerfiles inside employee-side-claim folder and hr-claim folder.
Command: vi docker-compose.yml

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:13
    container_name: postgres-claims
    ports:
      - "5801:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: admin123
      POSTGRES_DB: employee_claims_db
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      - claims-network

  backend:
    build: ./Backend
    container_name: backend-claims
    ports:
      - "3801:3801"
    depends_on:
      - postgres
    networks:
      - claims-network

  employee-side-claim:
    build: ./employee-side-claim
    container_name: employee-claim
    ports:
      - "8102:80"
    depends_on:
      - backend
    networks:
      - claims-network

  hr-claim:
    build: ./hr-claim
    container_name: hr-claim
    ports:
      - "8103:80"
    depends_on:
      - backend
    networks:
      - claims-network

volumes:
  pgdata:

networks:
  claims-network:
-- INSERT --
```

Then once written the code
Click on - Esc
Save the file = :wq (enter)

```
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ ls
Backend  docker-compose.yml  employee-side-claim  hr-claim  init.sql
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ cd Backend
ubuntu@ip-172-31-84-115:~/surekha-claims-management/Backend$ ls
node_modules  package-lock.json  package.json  server.js
ubuntu@ip-172-31-84-115:~/surekha-claims-management/Backend$
```

Command: cd Backend
Command: vi Dockerfile

```
FROM node:18

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3801

CMD ["node", "server.js"]

~
~
~
~
```

Esc
:wq (save) click on enter

Command: ls
Command: cd ..
Command: cd employee-sdie-claim
Command: ls
Command: vi Dockerfile

```
FROM nginx:alpine

COPY . /usr/share/nginx/html

EXPOSE 80

~
~
```

Esc
:wq

Command: ls
Command: cd ..
Command: cd hr-claim
Command: vi Dockerfile

```
FROM nginx:alpine

COPY . /usr/share/nginx/html

EXPOSE 80

~
~
```

Esc
:wq

Command: cd ..
Command: ls

Now finally to the run the servers
Command: docker-compose build & docker-compose up -d
or
Command: docker-compose up --build

Once you run the above command you will see like this

```
 ---> Running in def3ac0177d0
 ---> Removed intermediate container def3ac0177d0
 ---> 76ae0606d407
Step 7/7 : CMD ["node", "server.js"]
 ---> Running in ffdf1d60237a
 ---> Removed intermediate container ffdf1d60237a
 ---> 936849c83028
Successfully built 936849c83028
Successfully tagged surekha-claims-management_backend:latest
Building employee-side-claim
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  40.96kB
Step 1/3 : FROM nginx:alpine
 ---> 6769dc3a703c
Step 2/3 : COPY . /usr/share/nginx/html
 ---> 23844a2d7aa4
Step 3/3 : EXPOSE 80
 ---> Running in 47eb9f256bf6
 ---> Removed intermediate container 47eb9f256bf6
 ---> 0e24dff6557b
Successfully built 0e24dff6557b
Successfully tagged surekha-claims-management_employee-side-claim:latest
Building hr-claim
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  23.55kB
Step 1/3 : FROM nginx:alpine
 ---> 6769dc3a703c
Step 2/3 : COPY . /usr/share/nginx/html
 ---> 1368f21634cb
Step 3/3 : EXPOSE 80
 ---> Running in 63ed4de37029
 ---> Removed intermediate container 63ed4de37029
 ---> f96f714e4655
Successfully built f96f714e4655
Successfully tagged surekha-claims-management_hr-claim:latest
Creating network "surekha-claims-management_claims-network" with the default driver
Creating volume "surekha-claims-management_pgdata" with default driver
Creating postgres-claims ... done
Creating backend-claims  ... done
Creating employee-claim  ... done
Creating hr-claim        ... done
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ |
```

Once the containers are running

To check the background ports are running or not
Command: docker ps

To not running port means
Command; docker ps -a

```
Successfully tagged surekha-claims-management_hr-claim:latest
Creating network "surekha-claims-management_claims-network" with the default driver
Creating volume "surekha-claims-management_pgdata" with default driver
Creating postgres-claims ... done
Creating backend-claims  ... done
Creating employee-claim  ... done
Creating hr-claim        ... done
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS          PORTS
           NAMES
5d6edb8c4cd6   surekha-claims-management_employee-side-claim   "/docker-entrypoint.…"   37 minutes ago   Up 37 minutes   0.0.0.0:8102->80/tcp, [::]:8102->80/tcp
           employee-claim
4183562ebc60   surekha-claims-management_hr-claim             "/docker-entrypoint.…"   37 minutes ago   Up 37 minutes   0.0.0.0:8103->80/tcp, [::]:8103->80/tcp
           hr-claim
4644ca59a6a7   surekha-claims-management_backend              "docker-entrypoint.s…"   37 minutes ago   Up 37 minutes   0.0.0.0:3801->3801/tcp, :::3801->3801/tcp
           backend-claims
3c85909c8699   postgres:13                                    "docker-entrypoint.s…"   37 minutes ago   Up 37 minutes   0.0.0.0:5801->5432/tcp, [::]:5801->5432/t
cp         postgres-claims
```

To check the backend port take the container ID of backend port 3801

```
Creating employee-claim  ... done
Creating hr-claim        ... done
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS          PORTS
           NAMES
5d6edb8c4cd6   surekha-claims-management_employee-side-claim  "/docker-entrypoint.…"   37 minutes ago   Up 37 minutes   0.0.0.0:8102->80/tcp, [::]:8102->80/tcp
           employee-claim
4183562ebc60   surekha-claims-management_hr-claim             "/docker-entrypoint.…"   37 minutes ago   Up 37 minutes   0.0.0.0:8103->80/tcp, [::]:8103->80/tcp
           hr-claim
4644ca59a6a7   surekha-claims-management_backend              "docker-entrypoint.s…"   37 minutes ago   Up 37 minutes   0.0.0.0:3801->3801/tcp, :::3801->3801/tcp
           backend-claims
3c85909c8699   postgres:13                                    "docker-entrypoint.s…"   37 minutes ago   Up 37 minutes   0.0.0.0:5801->5432/tcp, [::]:5801->5432/t
cp         postgres-claims
```

Command: docker logs 4644ca59a6a7

Now check the posgres logs
Command: docker logs 3c85909c8699
Output: ready to accept database connections

To check the database is created or not
Command: docker exec -it 3c85909c8699 bash
Now login to postgres
Command: psql -U postgres
Command: psql

Now check the database is created or not
Command: \l
Output: employee_claims_db (this database name)

```
2025-06-12 07:47:08.289 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2025-06-12 07:47:08.296 UTC [64] LOG:  database system was shut down at 2025-06-12 07:47:08 UTC
2025-06-12 07:47:08.303 UTC [1] LOG:  database system is ready to accept connections
ubuntu@ip-172-31-84-115:~/surekha-claims-management$ docker exec -it 3c85909c8699 bash
root@3c85909c8699:/# psql -U postgres
psql (13.21 (Debian 13.21-1.pgdg120+1))
Type "help" for help.

postgres=# psql
postgres-# \l
                                  List of databases
        Name         |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
---------------------+----------+----------+------------+------------+-----------------------
 employee_claims_db | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres           | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0          | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                    |          |          |            |            | postgres=CTc/postgres
 template1          | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                    |          |          |            |            | postgres=CTc/postgres
(4 rows)

postgres-# \c employee_claims_db
You are now connected to database "employee_claims_db" as user "postgres".
employee_claims_db-# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | claims | table | postgres
(1 row)

employee_claims_db-# \d claims
                                      Table "public.claims"
        Column         |            Type             | Collation | Nullable |              Default
-----------------------+-----------------------------+-----------+----------+-------------------------------------
 id                    | integer                     |           | not null | nextval('claims_id_seq'::regclass)
 employee_id           | character varying(7)        |           | not null |
 employee_name         | character varying(50)       |           | not null |
 claim_type            | character varying(20)       |           | not null |
 claim_amount          | numeric(10,2)               |           | not null |
 status                | character varying(20)       |           | not null | 'Pending'::character varying
 submission_date       | timestamp without time zone |           | not null | CURRENT_TIMESTAMP
 hospital_name         | character varying(50)       |           |          |
 treatment_start_date  | date                        |           |          |
 treatment_end_date    | date                        |           |          |
```

To check the table inside the database
First connect to database
Command: \c employee_claims_db (enter)
To check the tables inside database
Command: \dt
Output: claims (table name)
To open the table

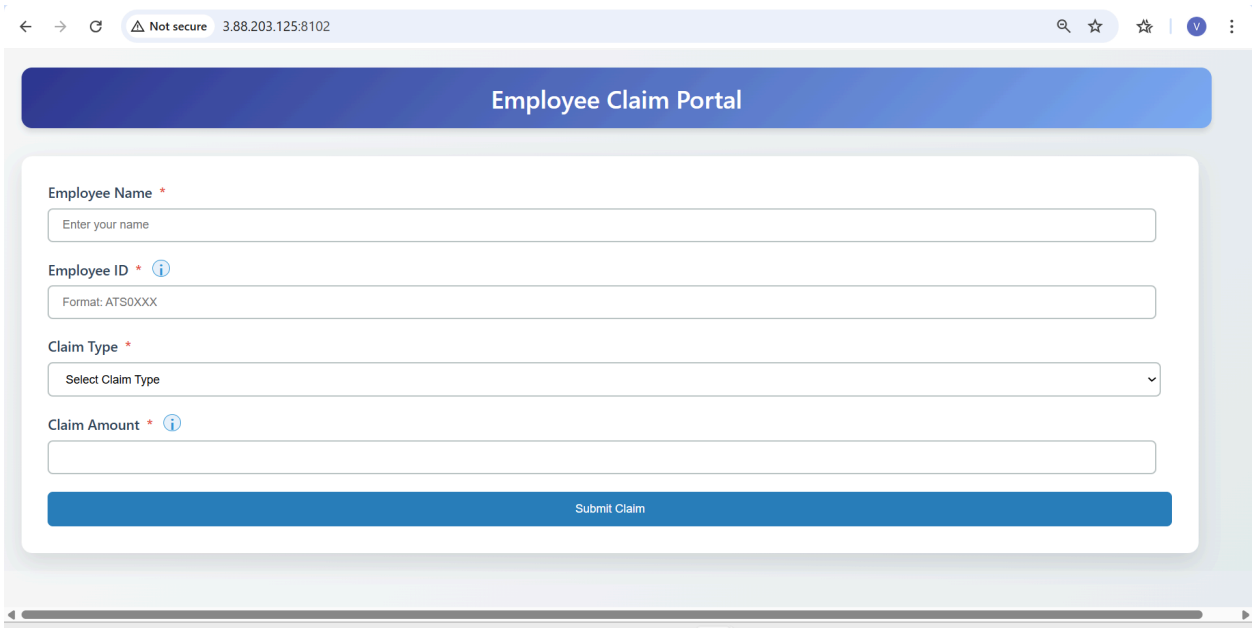
Command: \d claims

To quit from table
Command: \q

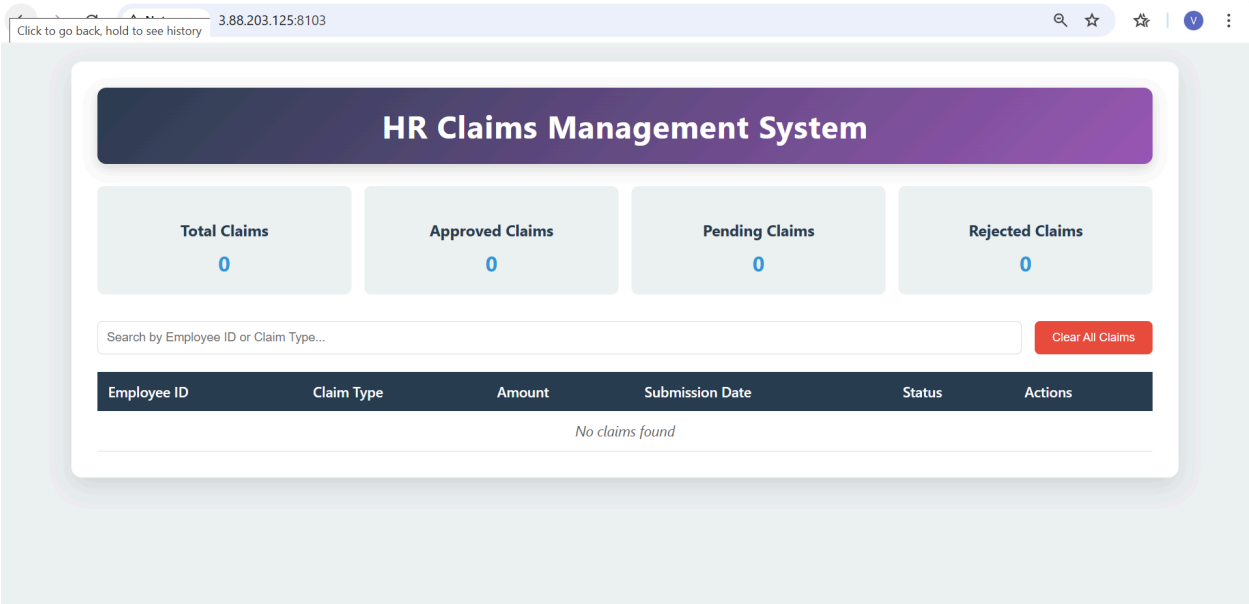
To exit from database
Command: exit
Output: back to host

Now Finally check the browser

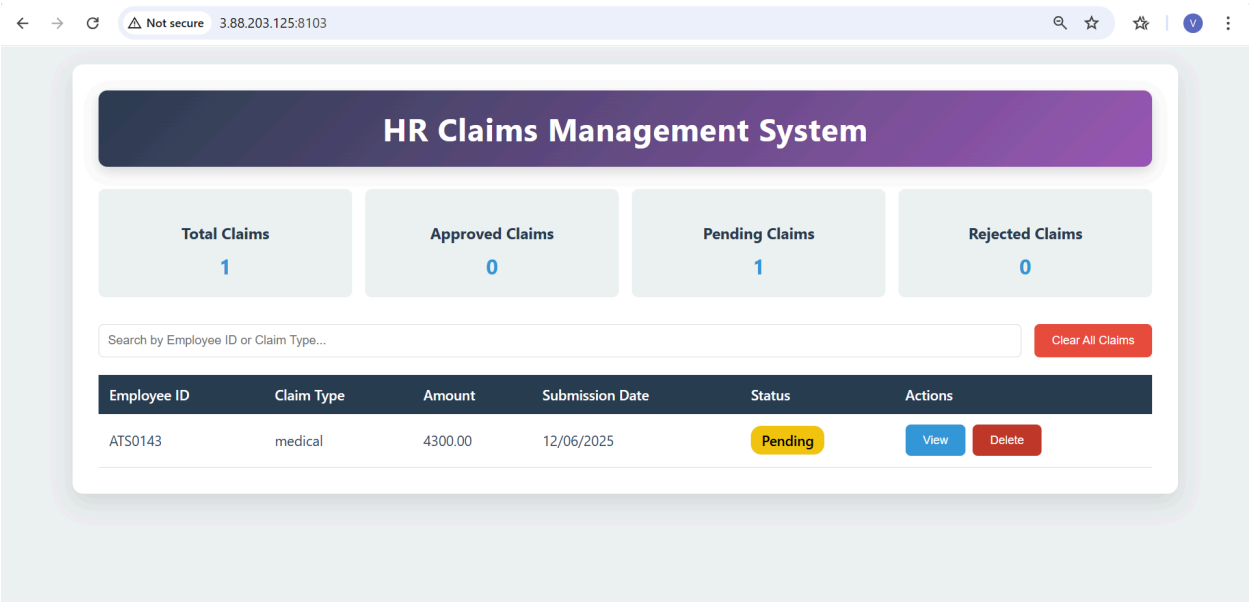
Now open your browser
Using ipaddress with port number
http://3.88.203.125:8102/



http://3.88.203.125:8103/



Enter the details and check



Update the URIs in sheet and give your name and give your github link after pushing to gitbash to github

Now to push this code to our github link
Commands: git status
Command: git add .
Command: git status (here you will see in green color which means added)
Command: git branch -M master or main (as per you branch in github)

Command: git push -u origin master or main
Command: give your github username
Command: paste your token link and click enter

Got back to your github account check code pushed to github

Note: to get your token link
Open github > click on username > select settings > developer settings > click on personal access token > token classic > generate token > select generate new token classic > give a name > select no expiry > tick repo box > click on create

Copy the link and go back to github and paste it and enter
Note: password will not be visible just paste and click on enter.

Step by step to create a New Server with installations first

Update your system
Command: sudo apt update && sudo apt upgrade -y

Install Docker:
Command: sudo apt install docker.io -y

Enable & start Docker
Command: sudo systemctl enable docker
Command: sudo systemctl start docker
Command: sudo systemctl status docker

Give Docker Permissions to current user
Command: sudo usermod -aG docker ubuntu

Command: logout

Again login into the server.

Now install Docker Compose

Command: sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Command: sudo chmod +x /usr/local/bin/docker-compose
Command: docker-compose --version

Install Postgresql

Command: sudo apt install postgresql postgresql-contrib -y

Start and Enable PostgreSQL
Command: sudo systemctl start postgresql
Command: sudo systemctl enable postgresql
Command: sudo systemctl status postgresql

Create PostgreSQL user

Command: sudo -u postgres psql

Inside the prompt, run:
Command:
CREATE USER postgres WITH PASSWORD 'admin123';
ALTER USER postgres WITH SUPERUSER;
\q

Note: above create user means creating postgres as user and giving password as admin123

so,
Now test the PostgreSQL Login

Command: psql -U postgres -h localhost

Note: if its ask password then admin123

If you get a connection error
Command: sudo nano /etc/postgresql/*/main/pg_hba.conf

Change this line : local all postgres peer

TO

To this : local all postgres md5

Then restart the postgreSQL
Command: sudo systemctl restart postgresql

Now install Node.js and npm

Command: sudo apt install nodejs npm -y
Command: node -v
Command: npm -v

Command: curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

Install node.js Dependencies

Command: npm install express cors multer dotenv path fs morgan helmet express-rate-limit
To check all the services status
Command: docker --version
Command: docker-compose --version
Command: systemctl status docker
Command: systemctl status postgresql
Command: psql -U postgres -h localhost
Command: node -v
Command: npm -v