1.index.js

```
import express from "express";
import cors from "cors";
import clientRoutes from "../src/routes/clientRoutes.js";

const app = express ();
const port = 3000;

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

app.get("/", (req, res) => {
  return res.send("<h1>Hello</h1>");
});

app.use("/api", clientRoutes);

app.listen(port, () => {
  console.log (`listening on port http://localhost:${port} `);
});
```

2.db.js

```
import pg from "pg";
import env from "dotenv";

env.config();
const db = new pg.Client({
  user: process.env.PG_USER,
  host: process.env.PG_HOST,
  database: process.env.PG_DATABASE,
  password: process.env.PG_PASSWORD,
  port: process.env.PG_PORT,
});

db.connect();

db.on("error", (err) => {
  console.error("unexpected error on the idle client", err);
  process.exit(-1);
});

export const query = (text, params) => {
  return db.query(text, params);
};
```

3.clientService.js

```
import {query} from "../db.js";

export const getClients = async () => {
  const { rows } = await query("SELECT * FROM public.clients_tb");
  return rows;
};


export const createClients = async (clientData) => {
  const { name, email, job, rate, isactive } = clientData;
  const { rows } = await query(
    `INSERT INTO clients_tb (name,email,job,rate,isactive) VALUES ($1,$2,$3,$4,$5) RETURNING
*`,
    [name, email, job, rate, isactive]
  );
  return rows[0];
};


export const updateClients = async (clientId, clientData) => {
  const { name, email, job, rate, isactive } = clientData;
  const { rows } = await query(
    `UPDATE  clients_tb SET name=$1,email=$2,job=$3,rate=$4,isactive=$5 WHERE id=$6
RETURNING * `,
    [name, email, job, rate, isactive, clientId]
  );
  return rows[0];
};
```

```
export const deleteClient = async (clientId) => {
  const { rowCount } = await query(`DELETE FROM clients_tb WHERE id = $1`, [
    clientId,
  ]);
  return rowCount > 0; // Returns true if a row was deleted, false otherwise
};

export const searchClients = async (searchTerm) => {
  const { rows } = await query(
    `SELECT * FROM clients_tb WHERE name ILIKE $1 OR email ILIKE $1 OR job ILIKE $1 `,
    [`%${searchTerm}%`]
  );
  return rows;
};
```

4. clientController.js

```
import * as clientService from "../services/clientServices.js";


export const getClients = async (req, res) => {
  try {
    const clients = await clientService.getClients();
    res.status(200).json(clients);
  } catch (err) {
    console.error("Error fetching clients:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};


export const createClients = async (req, res) => {
  try {
    const clientsData = req.body;
    const newClient = await clientService.createClients(clientsData);
    res.status(200).json(newClient);
  } catch (err) {
    console.error("Error adding client:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

```
export const updateClients = async (req, res) => {
  try {
    const clientId = req.params.id;
    const clientData = req.body;
    const updateClient = await clientService.updateClients(
      clientId,
      clientData
    );

    if (!updateClient) {
      return res.status(404).json({ message: "client not found" });
    }
    res.status(200).json(updateClient);
  } catch (err) {
    console.error("Error update clients:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};

export const deleteClients = async (req, res) => {
  try {
    const clientId = req.params.id;
    const deleteClient = await clientService.deleteClient(clientId);

    if (!deleteClient) {
      return res.status(404).json({ message: "client not found" });
```

```
    }
    res.status(200).json(deleteClient);
  } catch (err) {
    console.error("Error delete client:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};

export const searchClients = async (req, res) => {
  try {
    const searchTerm = req.query.q;
    const clients = await clientService.searchClients(searchTerm);
    res.status(200).json(clients);
  } catch (err) {
    console.error("Error searching clients:", err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

5. clientRoutes.js

```
import express from "express";


import * as clientController from "../controllers/clientController.js";


const router = express.Router();


router.get("/clients", clientController.getClients);
router.post("/clients", clientController.createClients);
router.put("/clients/:id", clientController.updateClients);
router.delete("/clients/:id", clientController.deleteClients);
router.get("/clients/search", clientController.searchClients);
export default router;
```