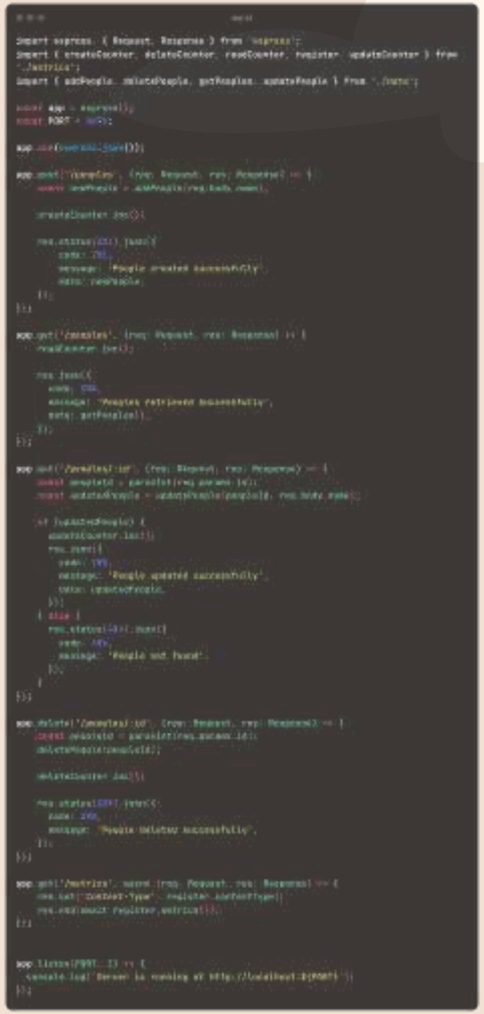
Monitor Node with Prometheus + Grafana

Application Monitoring

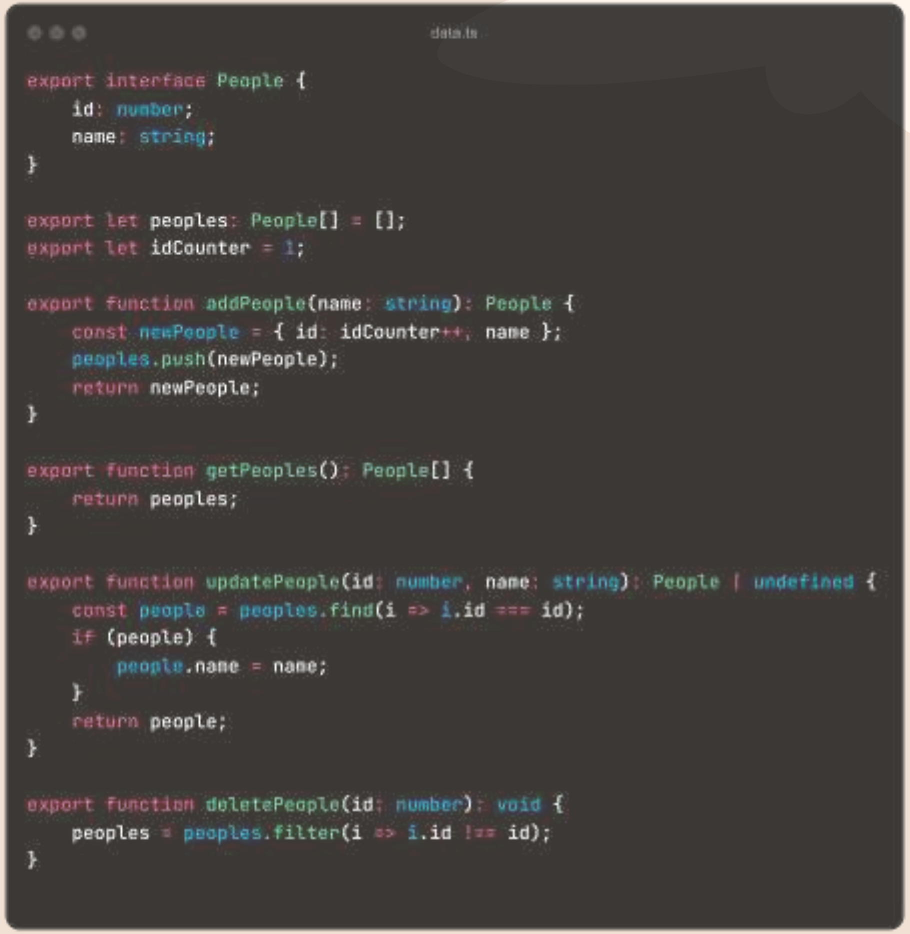
Create a file named app.ts

First, create a Node.js application using the Express framework with a file named app.ts



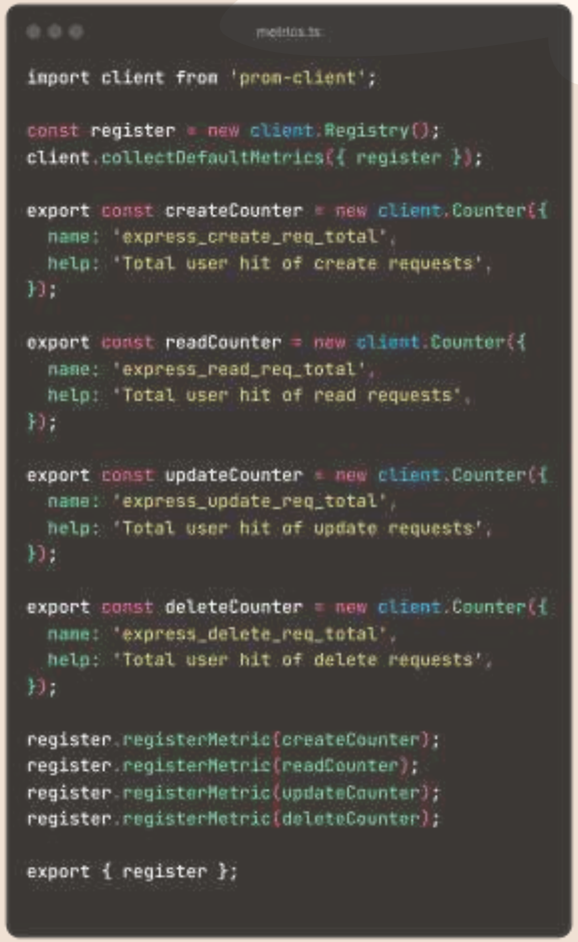
```typescript
import express, { Request, Response } from 'express';
import { createCounter, deleteCounter, readCounter, register, updateCounter } from
'./metrics';
import { addPeople, deletePeople, getPeoples, updatePeople } from './data';

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

app.use(express.json());

app.post('/peoples', (req: Request, res: Response) => {
  const newPeople = addPeople(req.body.name);

  createCounter.inc();

  res.status(201).json({
    code: 201,
    message: 'People created successfully',
    data: newPeople,
  });
});

app.get('/peoples', (req: Request, res: Response) => {
  readCounter.inc();

  res.json({
    code: 200,
    message: 'Peoples retrieved successfully',
    data: getPeoples(),
  });
});

app.put('/peoples/:id', (req: Request, res: Response) => {
  const peopleId = parseInt(req.params.id);
  const updatedPeople = updatePeople(peopleId, req.body.name);

  if (updatedPeople) {
    updateCounter.inc();
    res.json({
      code: 200,
      message: 'People updated successfully',
      data: updatedPeople,
    });
  } else {
    res.status(404).json({
      code: 404,
      message: 'People not found',
    });
  }
});

app.delete('/peoples/:id', (req: Request, res: Response) => {
  const peopleId = parseInt(req.params.id);
  deletePeople(peopleId);

  deleteCounter.inc();

  res.status(200).json({
    code: 200,
    message: 'People deleted successfully',
  });
});

app.get('/metrics', async (req: Request, res: Response) => {
  res.set('Content-Type', register.contentType);
  res.end(await register.metrics());
});


app.listen(PORT, () => {
  console.log(`Server is running at http://localhost:${PORT}`);
});
```

Create a file named data.ts



```typescript
export interface People {
    id: number;
    name: string;
}

export let peoples: People[] = [];
export let idCounter = 1;

export function addPeople(name: string): People {
    const newPeople = { id: idCounter++, name };
    peoples.push(newPeople);
    return newPeople;
}

export function getPeoples(): People[] {
    return peoples;
}

export function updatePeople(id: number, name: string): People | undefined {
    const people = peoples.find(i => i.id === id);
    if (people) {
        people.name = name;
    }
    return people;
}

export function deletePeople(id: number): void {
    peoples = peoples.filter(i => i.id !== id);
}
```

Create a file named metrics.ts



```
import client from 'prom-client';

const register = new client.Registry();
client.collectDefaultMetrics({ register });

export const createCounter = new client.Counter({
  name: 'express_create_req_total',
  help: 'Total user hit of create requests',
});

export const readCounter = new client.Counter({
  name: 'express_read_req_total',
  help: 'Total user hit of read requests',
});

export const updateCounter = new client.Counter({
  name: 'express_update_req_total',
  help: 'Total user hit of update requests',
});

export const deleteCounter = new client.Counter({
  name: 'express_delete_req_total',
  help: 'Total user hit of delete requests',
});

register.registerMetric(createCounter);
register.registerMetric(readCounter);
register.registerMetric(updateCounter);
register.registerMetric(deleteCounter);

export { register };
```

GRAFANA MOCKUP

SEE YOU

●●●●●●