```
npm install express-rate-limit
```

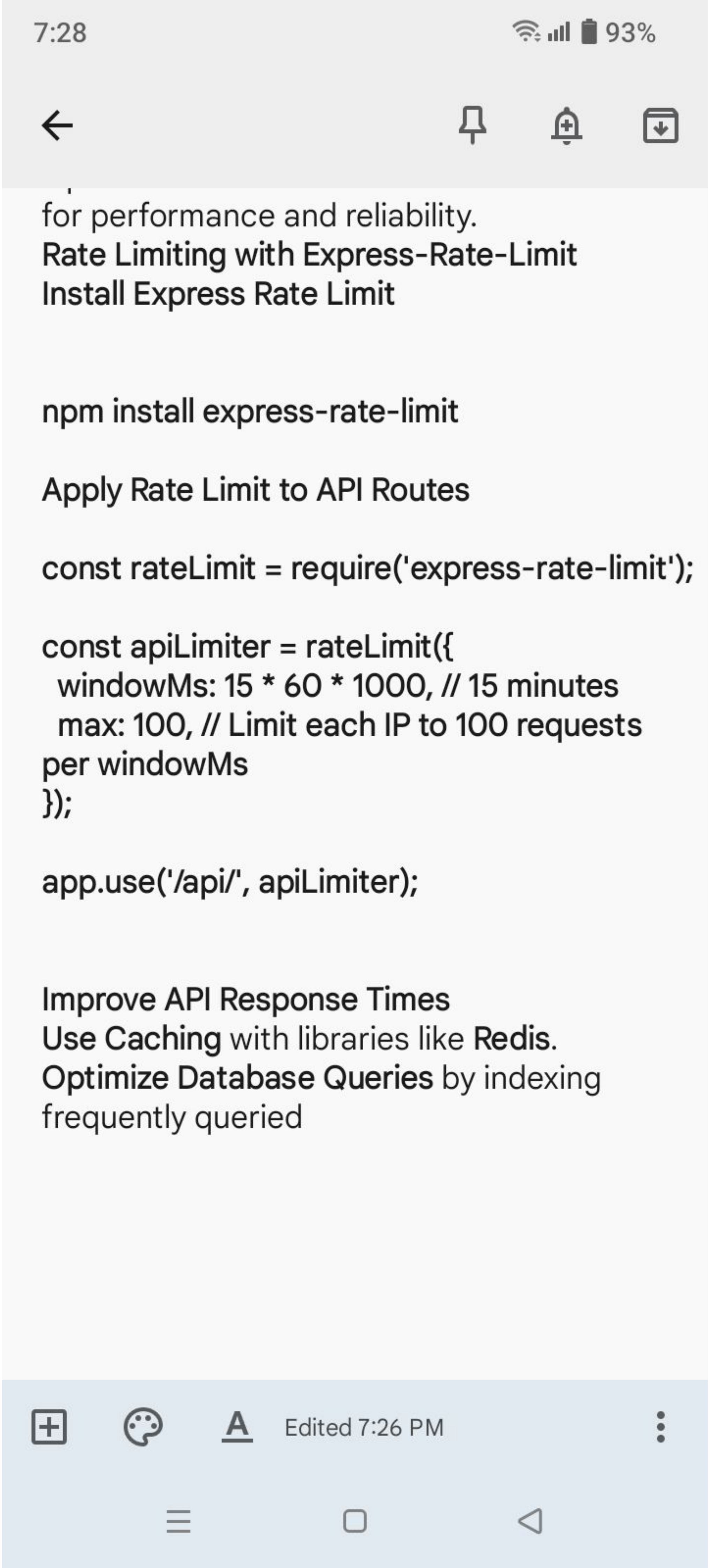
Apply Rate Limit to API Routes

```
const rateLimit = require('express-rate-limit');

const apiLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100, // Limit each IP to 100 requests
per windowMs
});

app.use('/api/', apiLimiter);
```

Improve API Response Times

Use Caching with libraries like **Redis.**

Optimize Database Queries by indexing frequently queried



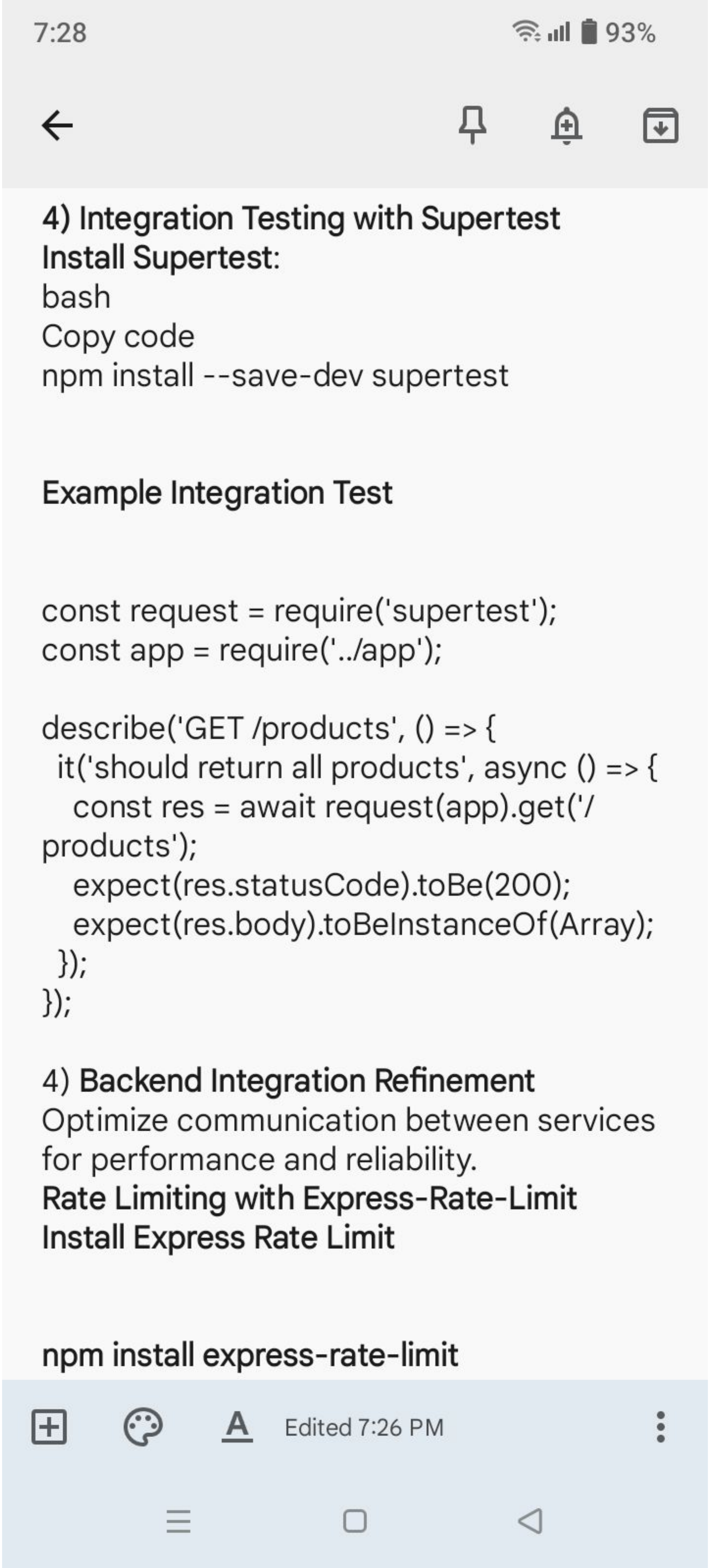
for performance and reliability.
Rate Limiting with Express-Rate-Limit
Install Express Rate Limit

```
npm install express-rate-limit
```

Apply Rate Limit to API Routes

```
const rateLimit = require('express-rate-limit');

const apiLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100, // Limit each IP to 100 requests
per windowMs
});

app.use('/api/', apiLimiter);
```

Improve API Response Times
Use Caching with libraries like **Redis**.
Optimize Database Queries by indexing frequently queried



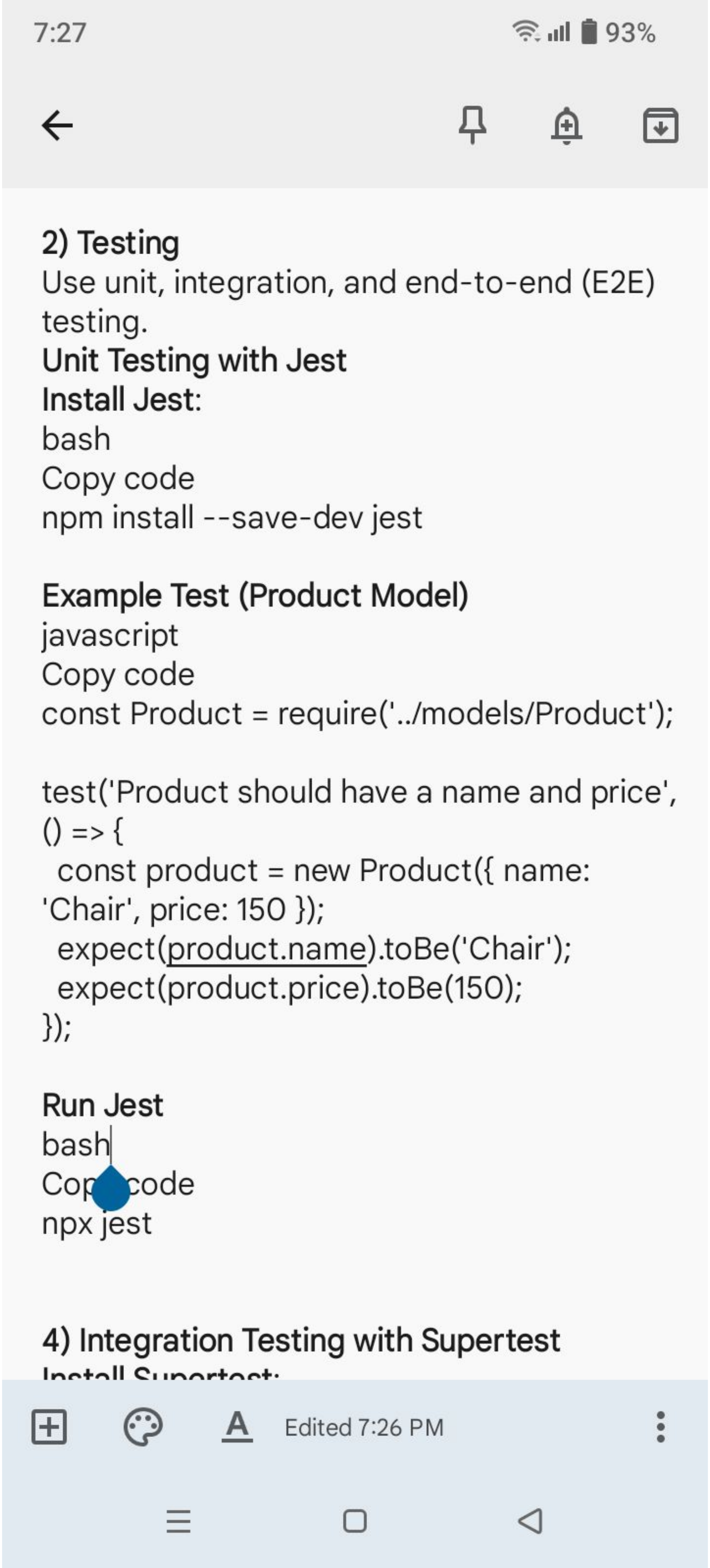
4) Integration Testing with Supertest
Install Supertest:
bash
Copy code
```
npm install --save-dev supertest
```

Example Integration Test

```
const request = require('supertest');
const app = require('../app');

describe('GET /products', () => {
  it('should return all products', async () => {
    const res = await request(app).get('/products');
    expect(res.statusCode).toBe(200);
    expect(res.body).toBeInstanceOf(Array);
  });
});
```

4) **Backend Integration Refinement**
Optimize communication between services for performance and reliability.
Rate Limiting with Express-Rate-Limit
Install Express Rate Limit

```
npm install express-rate-limit
```

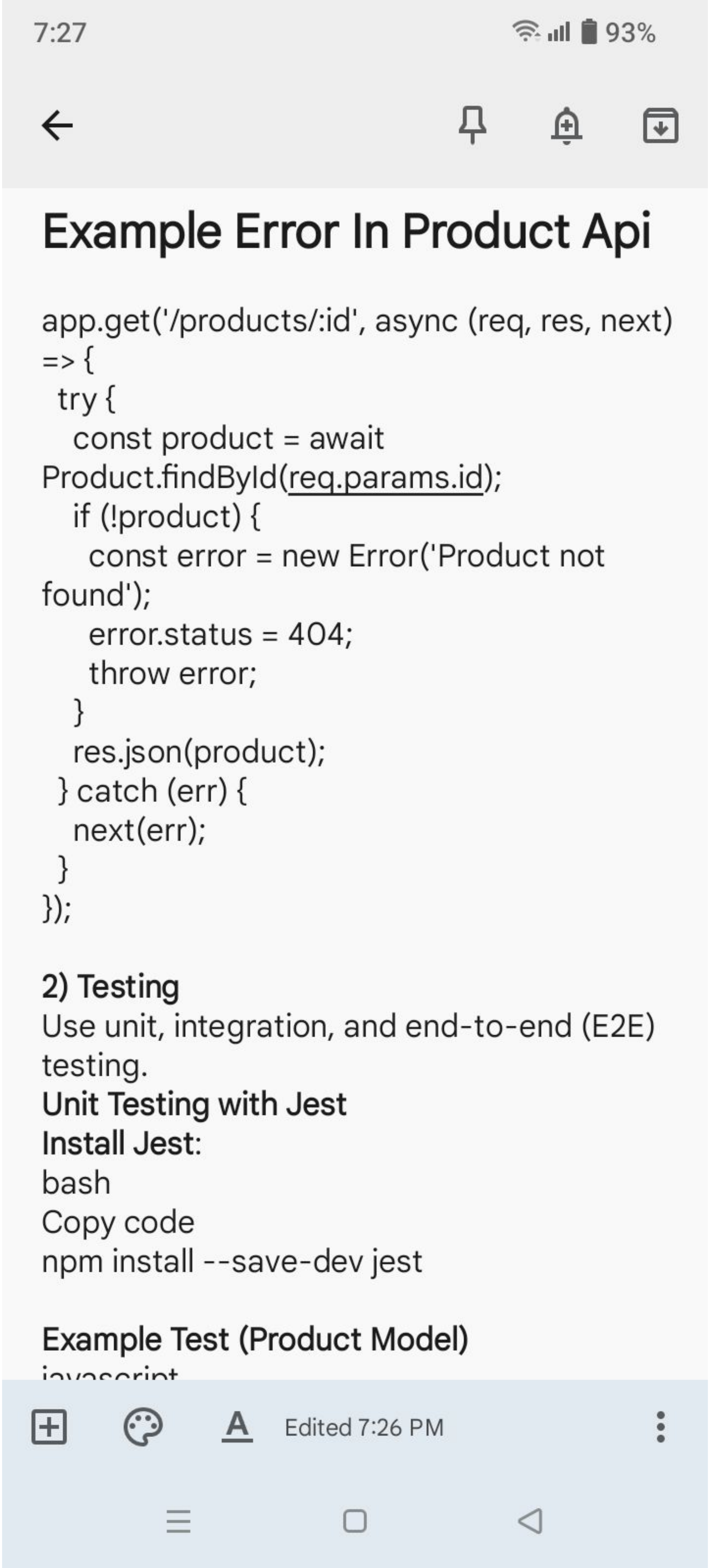


2) Testing
Use unit, integration, and end-to-end (E2E) testing.
Unit Testing with Jest
Install Jest:
bash
Copy code
```
npm install --save-dev jest
```

Example Test (Product Model)
javascript
Copy code
```
const Product = require('../models/Product');

test('Product should have a name and price', () => {
  const product = new Product({ name: 'Chair', price: 150 });
  expect(product.name).toBe('Chair');
  expect(product.price).toBe(150);
});
```

Run Jest
bash
Copy code
```
npx jest
```

4) Integration Testing with Supertest
Install Supertest:

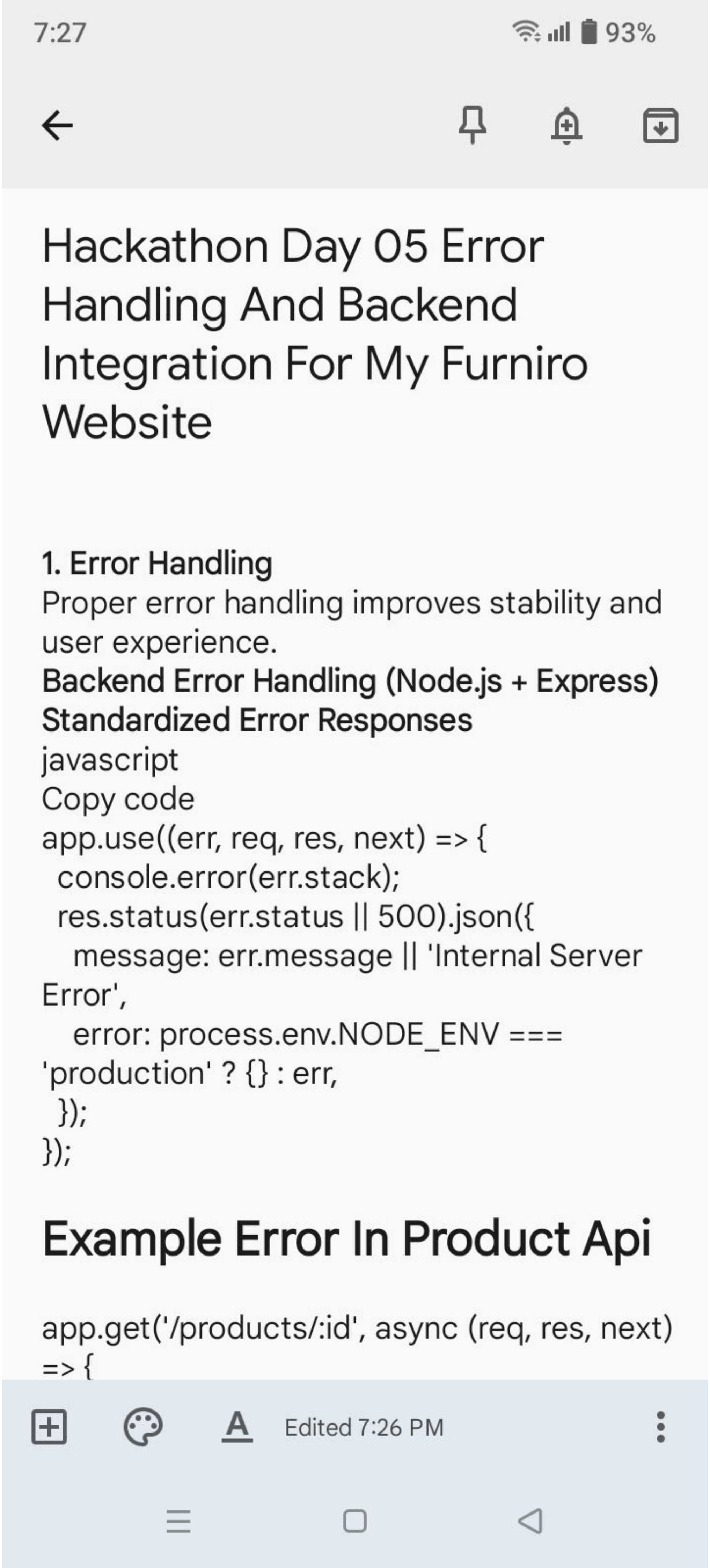


Example Error In Product Api

```
app.get('/products/:id', async (req, res, next)
=> {
  try {
    const product = await
Product.findById(req.params.id);
    if (!product) {
      const error = new Error('Product not
found');
      error.status = 404;
      throw error;
    }
    res.json(product);
  } catch (err) {
    next(err);
  }
});
```

2) Testing
Use unit, integration, and end-to-end (E2E) testing.
Unit Testing with Jest
Install Jest:
bash
Copy code
```
npm install --save-dev jest
```

Example Test (Product Model)
javascript



Hackathon Day 05 Error Handling And Backend Integration For My Furniro Website

1. Error Handling
Proper error handling improves stability and user experience.
Backend Error Handling (Node.js + Express)
Standardized Error Responses
javascript
Copy code

```javascript
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({
    message: err.message || 'Internal Server Error',
    error: process.env.NODE_ENV === 'production' ? {} : err,
  });
});
```

Example Error In Product Api

```javascript
app.get('/products/:id', async (req, res, next) => {
```