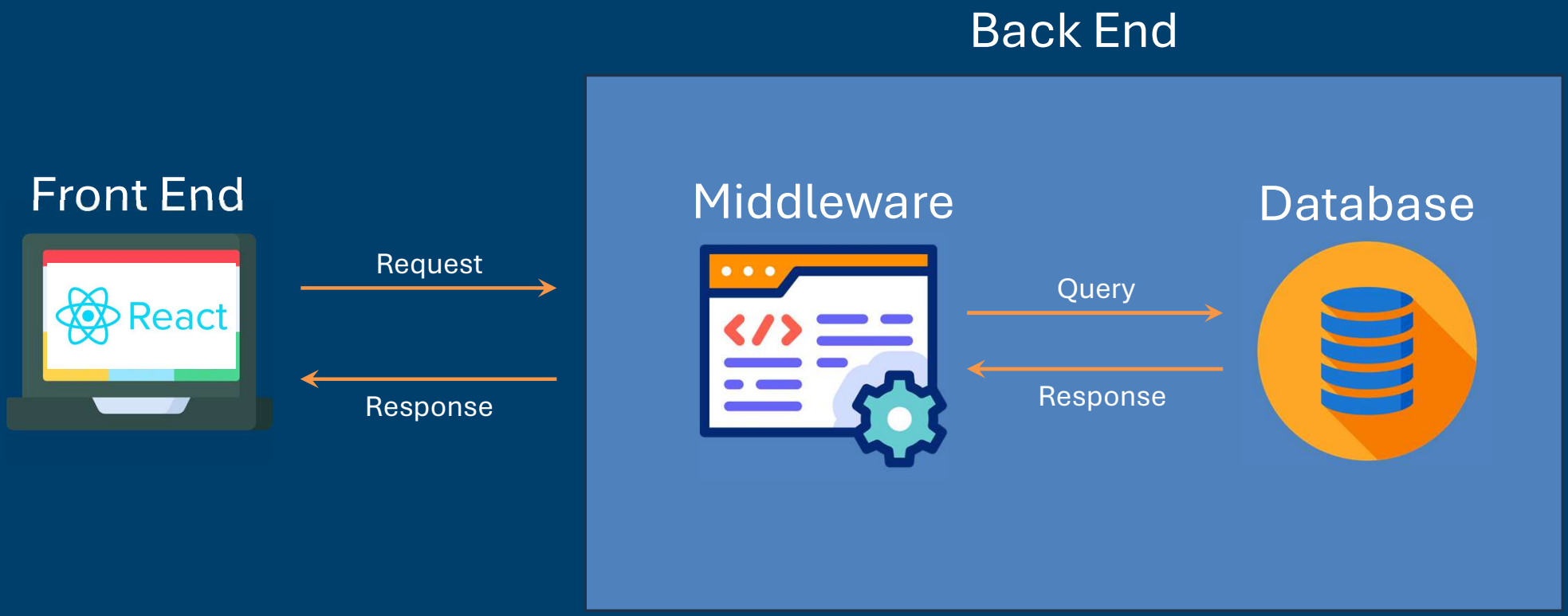
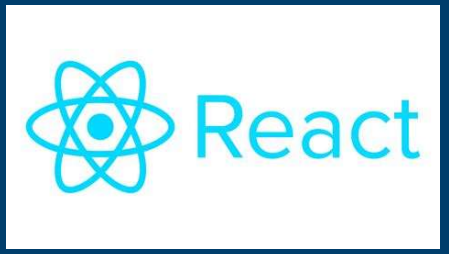
Creating a Database Backend with Prisma and Postgres


Back End
Front End
Middleware
Database
React
Request
Response
Query
Response

Front End

- React
- Alternative interfaces



Middleware

- Node.js
- Express
- Prisma

Prisma

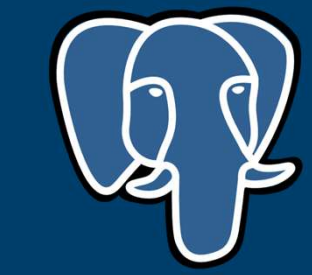
- Object-relational mapping (ORM)
- TypeScript -> SQL Querying
- Great for integration with REACT
- Node.js with Express

Database

- Docker
- PostgreSQL

Postgres

- Object-Relational Database Management System
- Uses and extends the SQL Language
- ACID-compliant
- Free and Open Source



PostgreSQL

Front End

React

Middleware

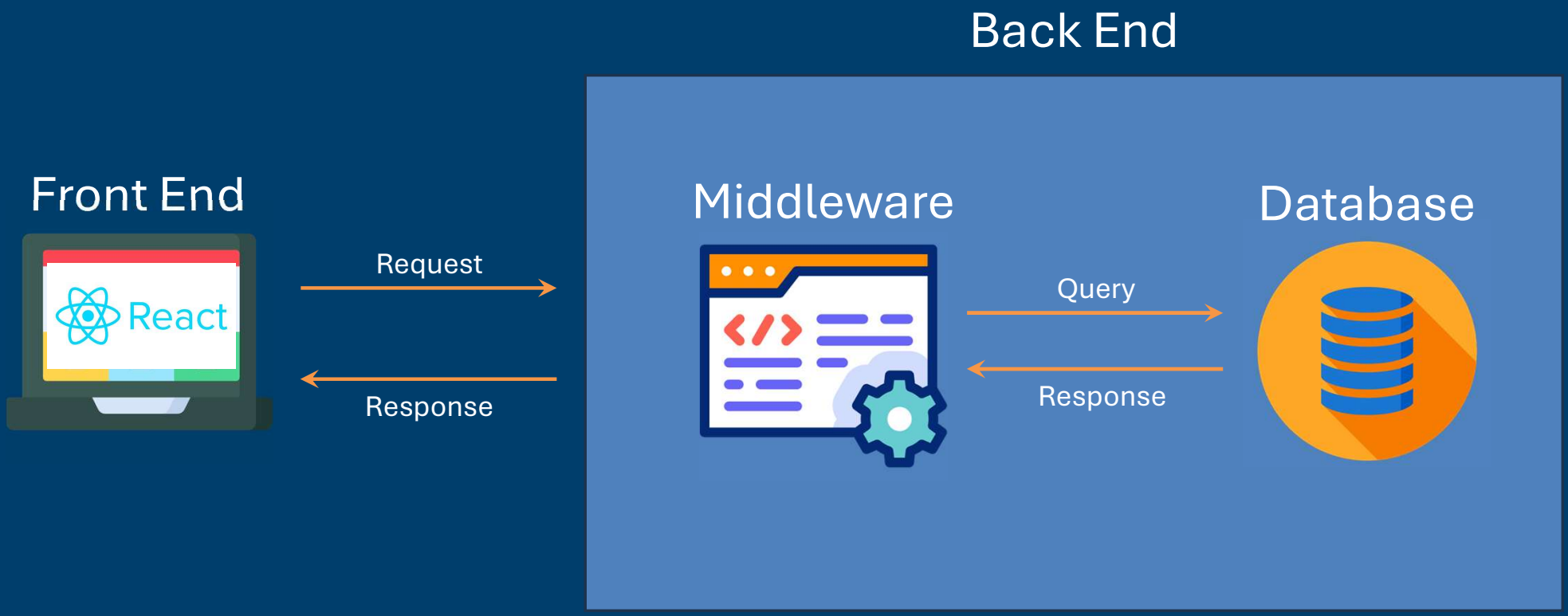
node express

Prisma

Database

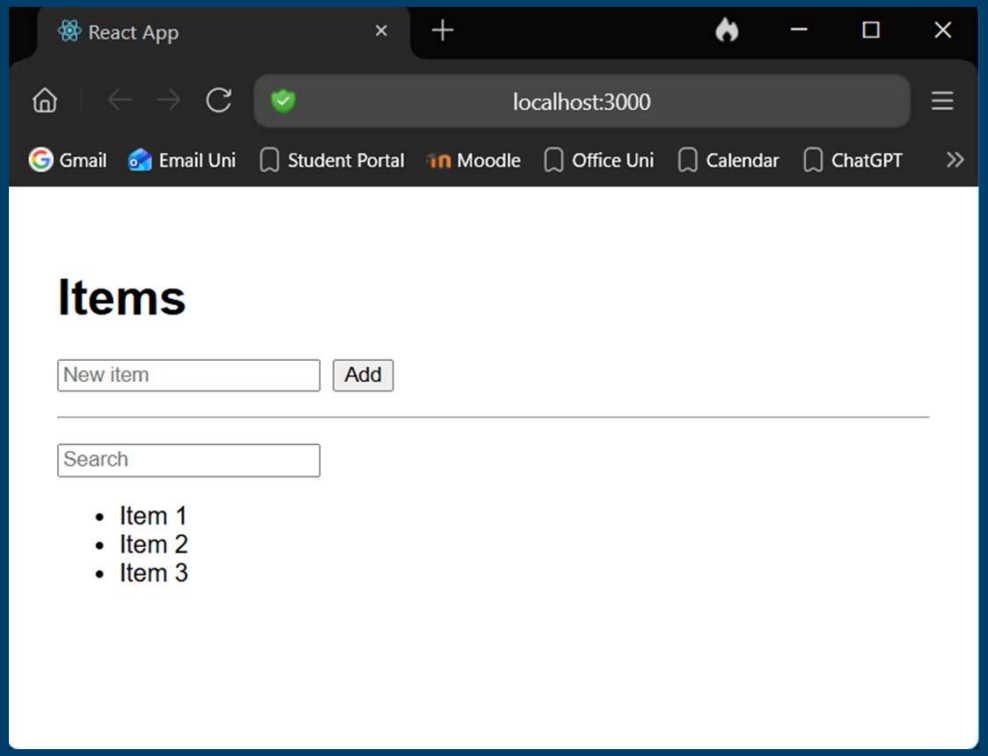
docker

PostgreSQL

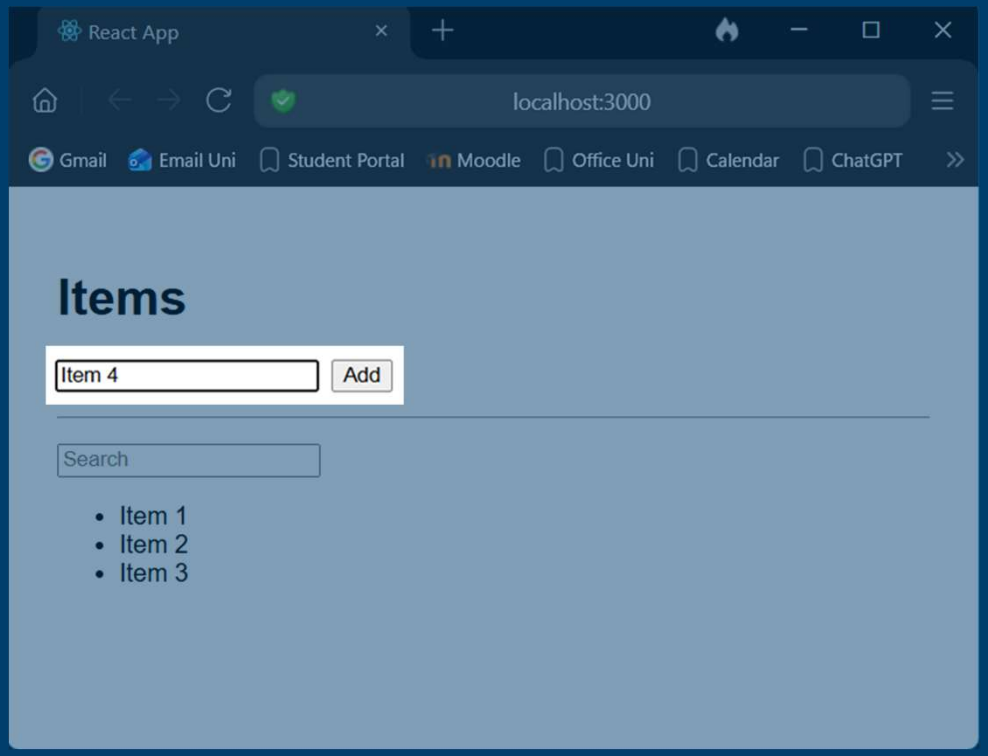

Back End
Front End
Middleware
Database
React
Request
Response
Query
Response

Tutorial

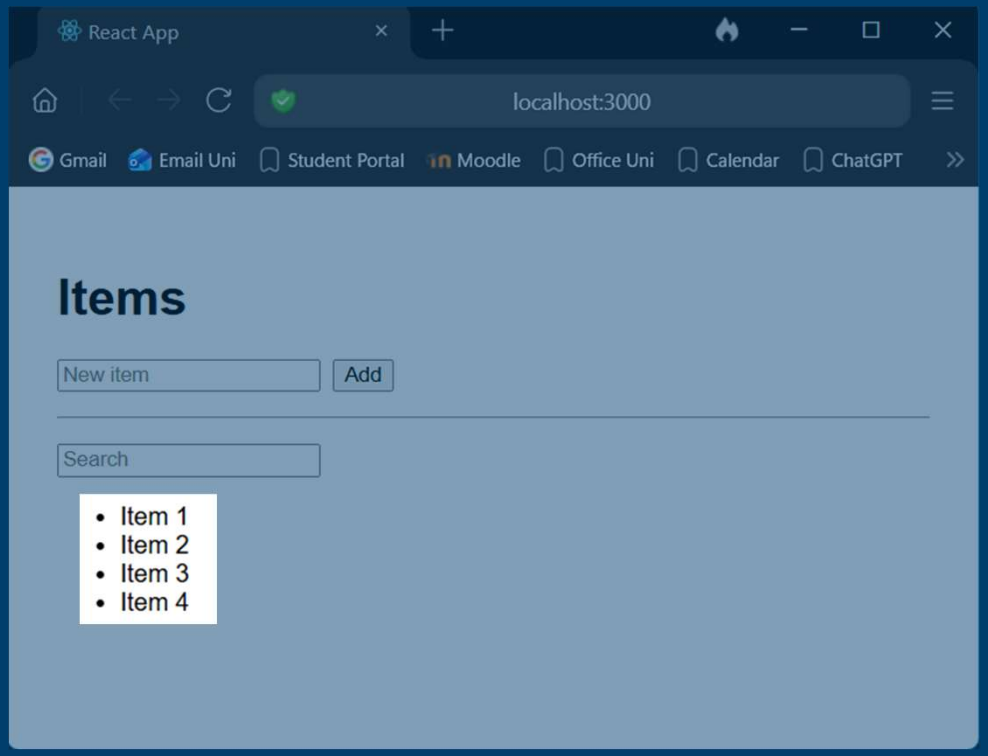
The Project


React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
New item
Add
Search
Item 1
Item 2
Item 3

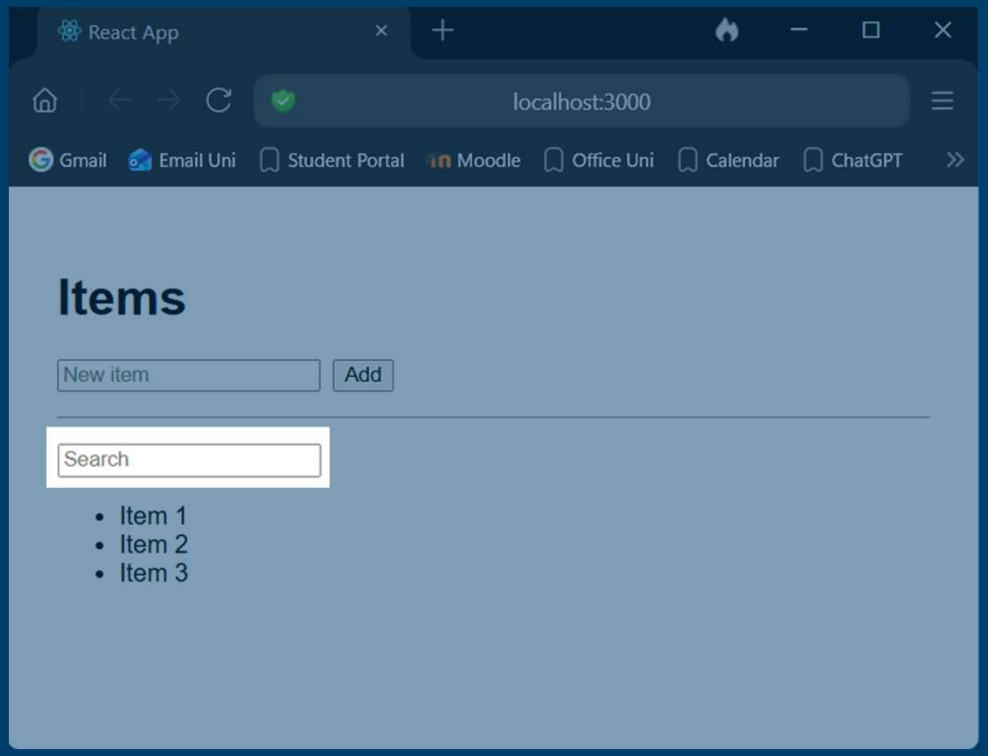
The Project


React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
Item 4
Add
Search
Item 1
Item 2
Item 3

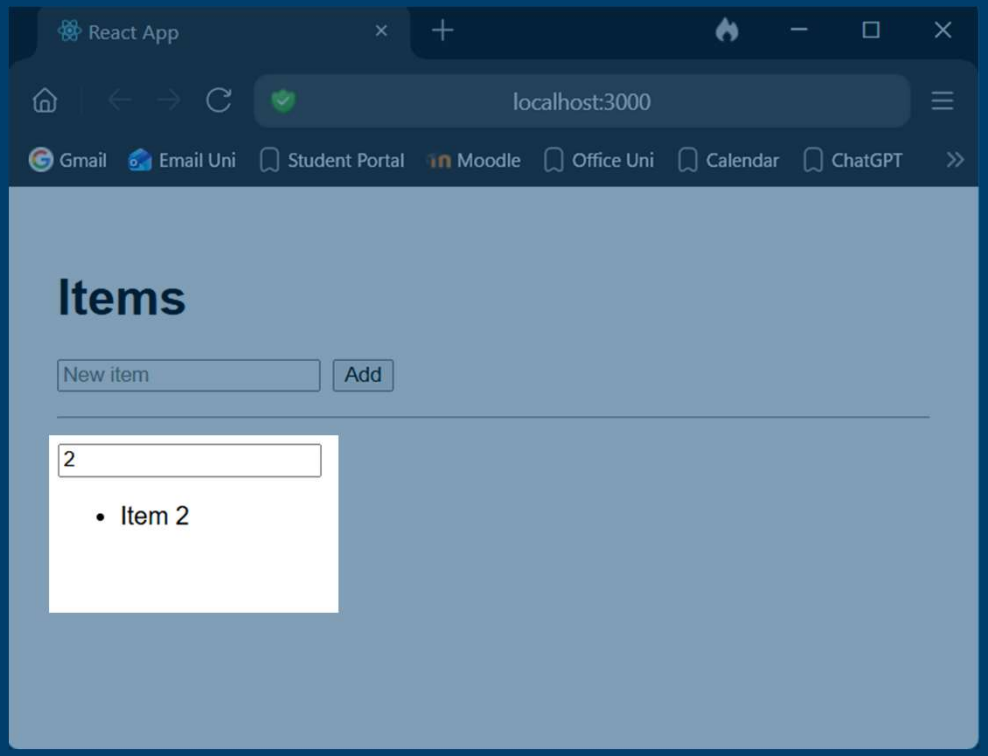
The Project


React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
New item
Add
Search
Item 1
Item 2
Item 3
Item 4

The Project

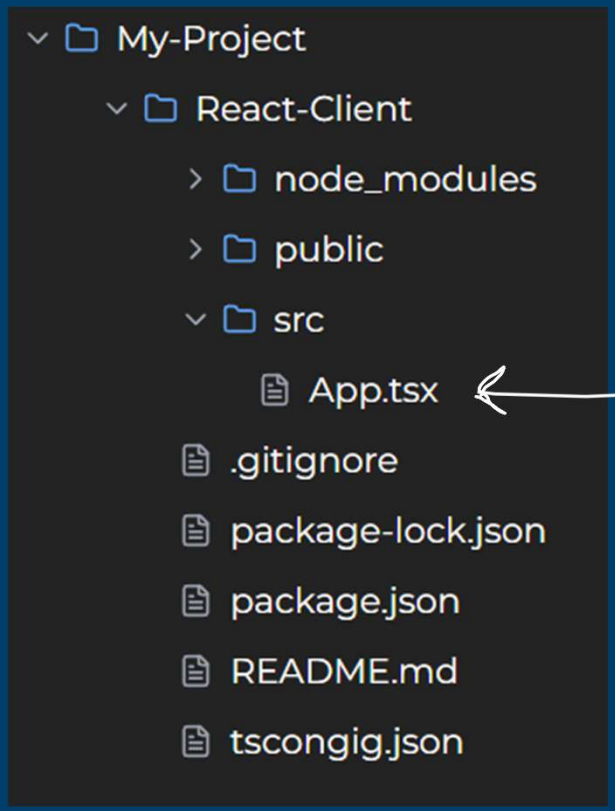

React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
New item
Add
Search
Item 1
Item 2
Item 3

The Project


React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
New item
Add
2
Item 2

The Project

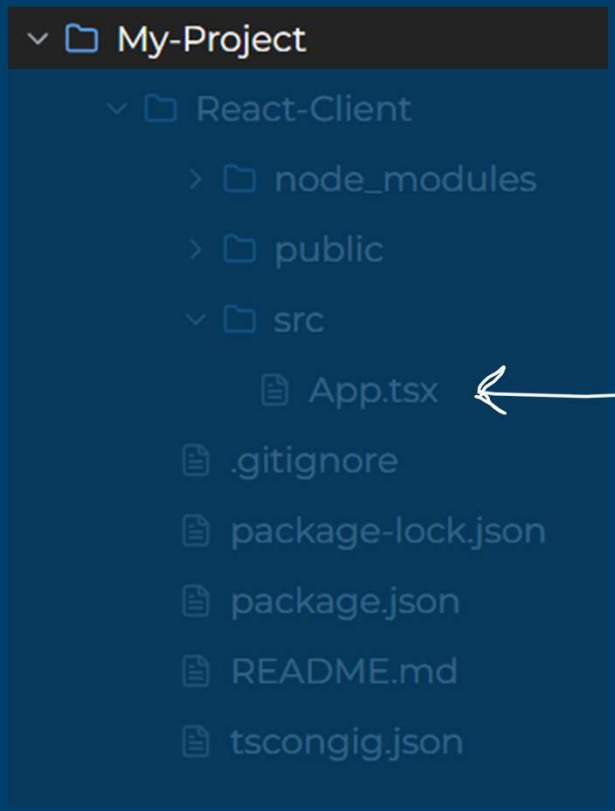
```
npx create-react-app React-Client --template typescript
```



Only this file has been edited

The Project

```
npx create-react-app React-Client --template typescript
```



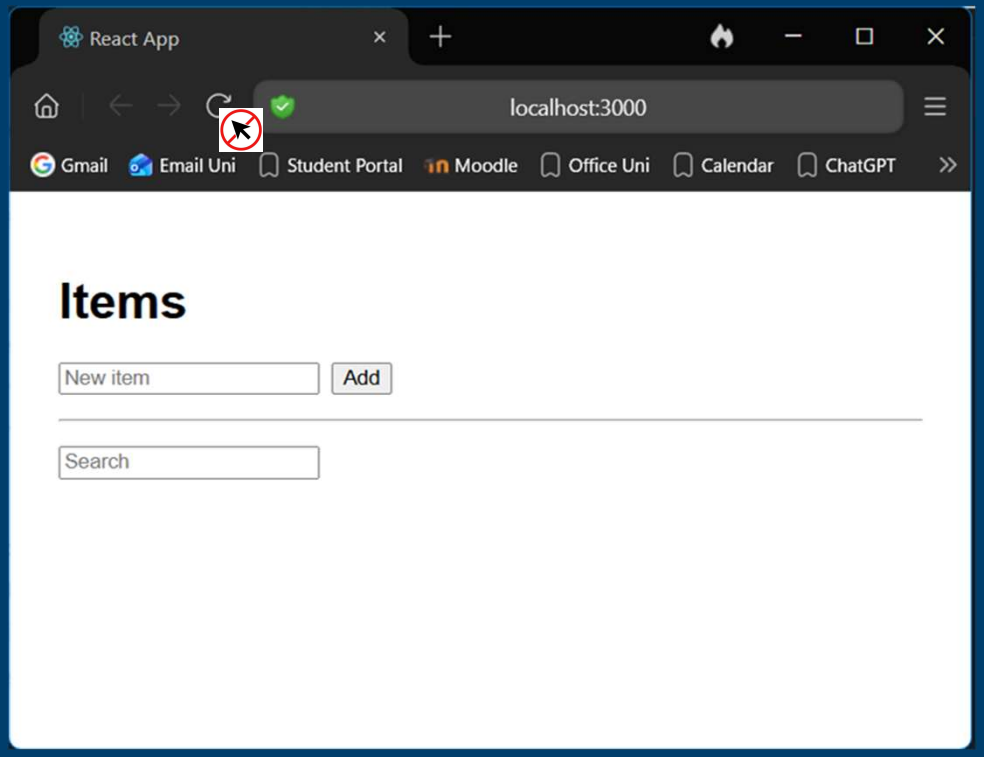
Only this file has been edited

The Project - ./React-Client/src/App.tsx

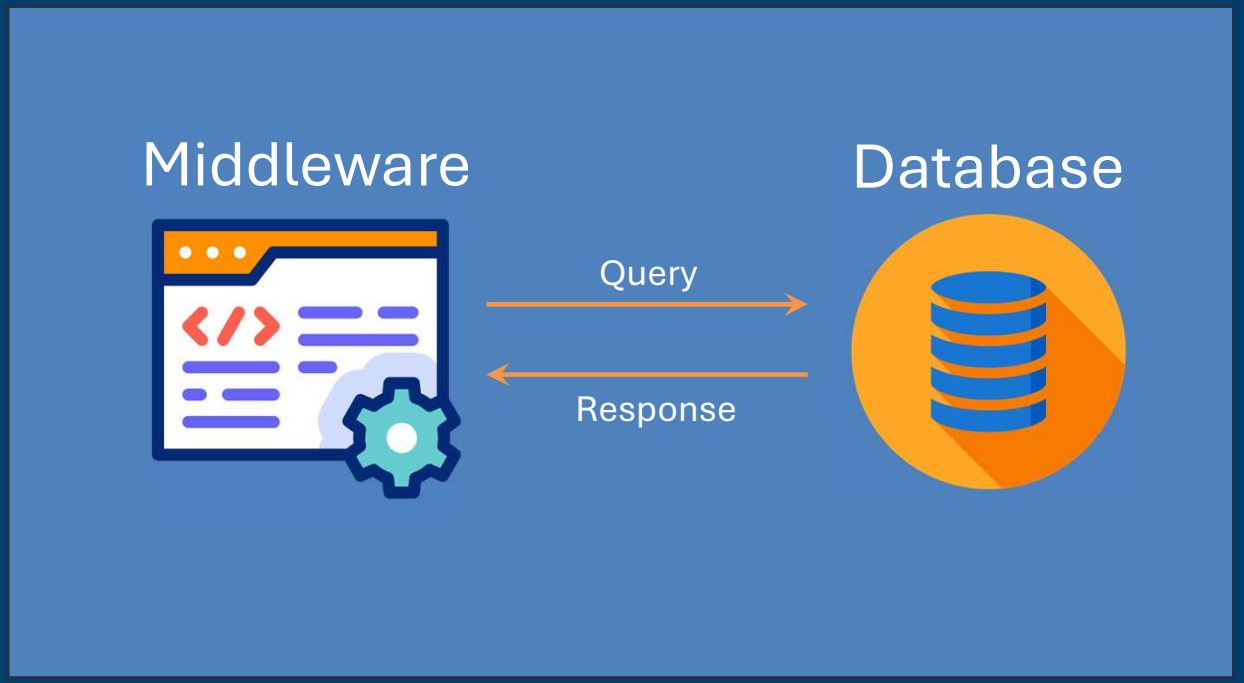
```tsx
import { useState } from "react";

interface Item {
  id: number;
  name: string;
}

function App() {
  const [items, setItems] = useState<Item[]>([]);
  const [name, setName] = useState("");
  const [search, setSearch] = useState("");

  const createItem = () => {
    if (!name.trim()) return;

    const newItem: Item = {
      id: items.length ? items.at(-1)!.id + 1 : 1,
      name,
    };
    setItems([...items, newItem]);
    setName("");
  };

  const searchItems = (q: string) => {
    setSearch(q);
  };

  const filteredItems = items.filter((item) =>
    item.name.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <div style={{ padding: "2rem", fontFamily: "sans-serif" }}>
      <h1>Items</h1>

      <input
        placeholder="New item"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <button onClick={createItem} style={{ marginLeft: "0.5rem" }}>
        Add
      </button>

      <hr style={{ margin: "1rem 0" }} />

      <input
        placeholder="Search"
        value={search}
        onChange={(e) => searchItems(e.target.value)}
      />

      <ul>
        {filteredItems.map((item) => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

The problem...

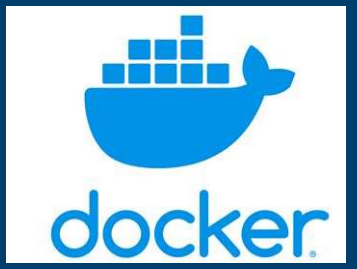

NO


React App
localhost:3000
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
ChatGPT
Items
New item
Add
Search

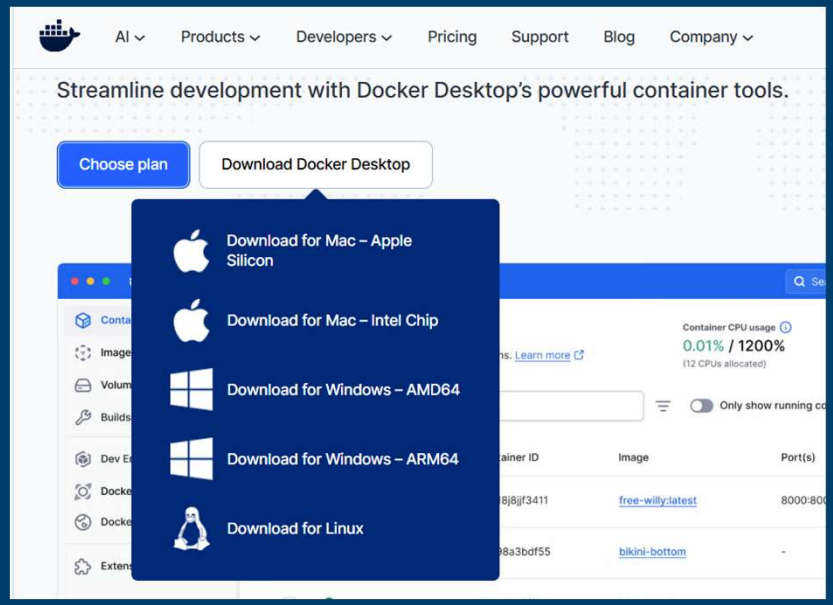
Back End

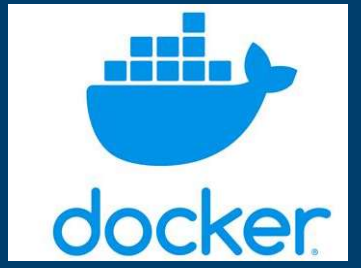


Creating Postgres Database

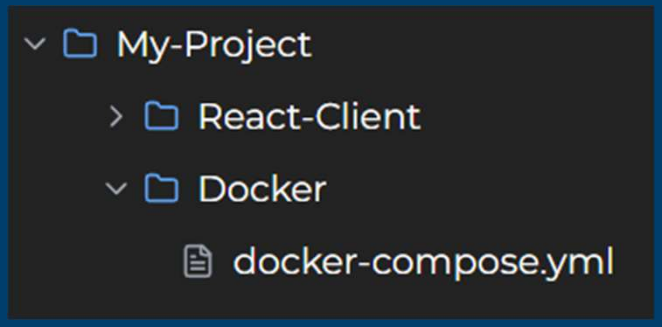


- Docker Desktop from www.docker.com

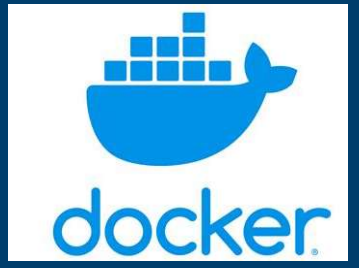




- Create a Docker folder and docker-compose.yml

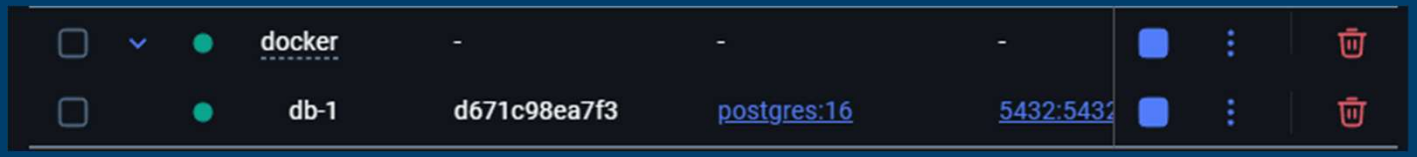


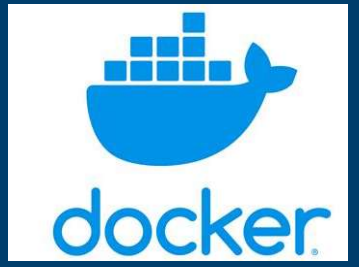

```yaml
version: '3.9'

services:
  db:
    image: postgres:16
    restart: always
    environment:
      POSTGRES_USER: app_user
      POSTGRES_PASSWORD: app_password
      POSTGRES_DB: app_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```



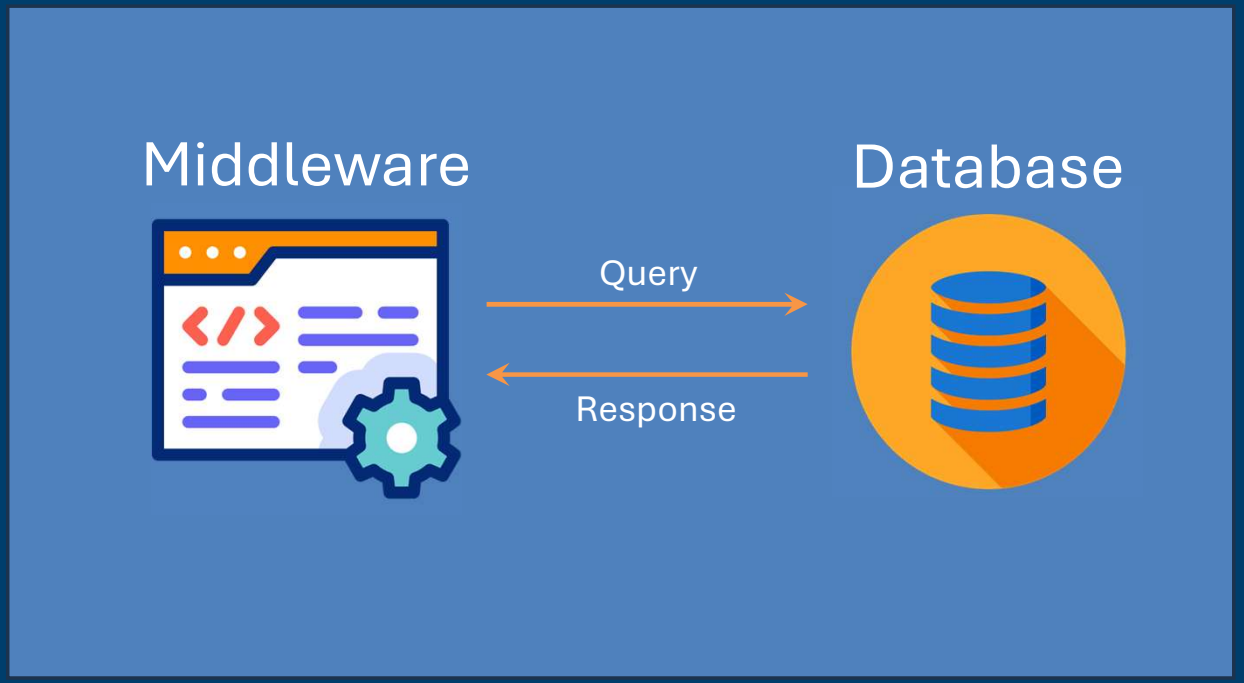
- Run the command to start Postgres

```
cd .\Docker\
docker compose up -d
```



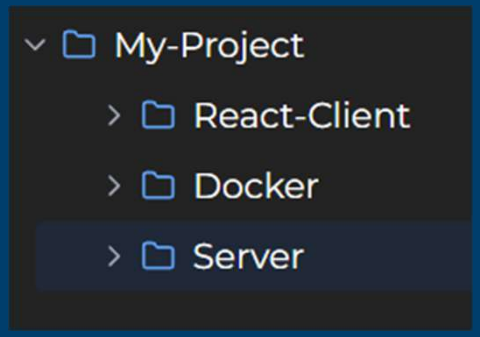


Back End



Initialising Node.js Middleware

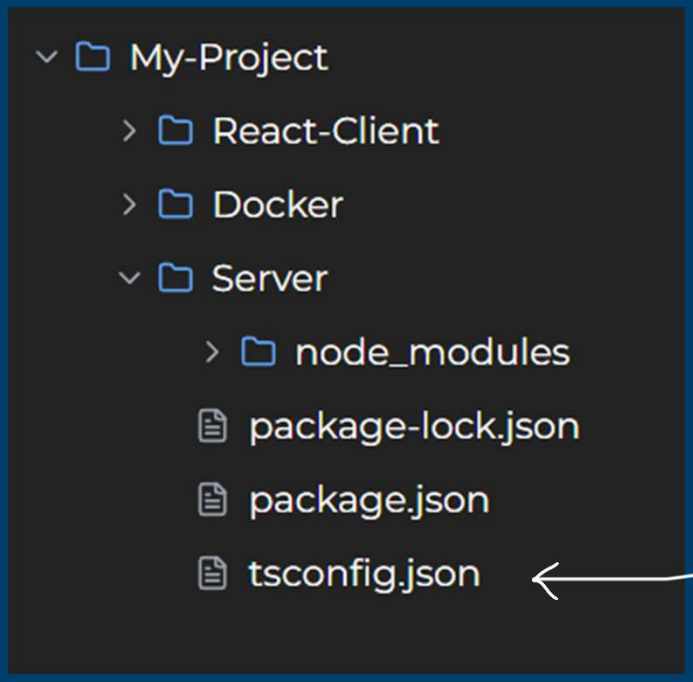
- Create a Server directory



- Initialise Node.js

```
cd .\Server\
npm init -y
npm install express cors
npm install -D typescript ts-node nodemon @types/node @types/express
@types/cors
npx tsc --init
```

- TypeScript Configuration




```
{
  "compilerOptions": {
    "module": "CommonJS",
    "target": "ES2020",
    "moduleResolution": "Node",
    "strict": true,
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "noEmit": true,
    "allowImportingTsExtensions": true,
    "rootDir": "./src",
    "outDir": "./dist",
  },
  "include": ["src/index.ts", "prisma/**/*.ts"]
}
```

Configuring Prisma

```
npm install -D prisma@6.19.0 @prisma/client@6.19.0
npx prisma init
```

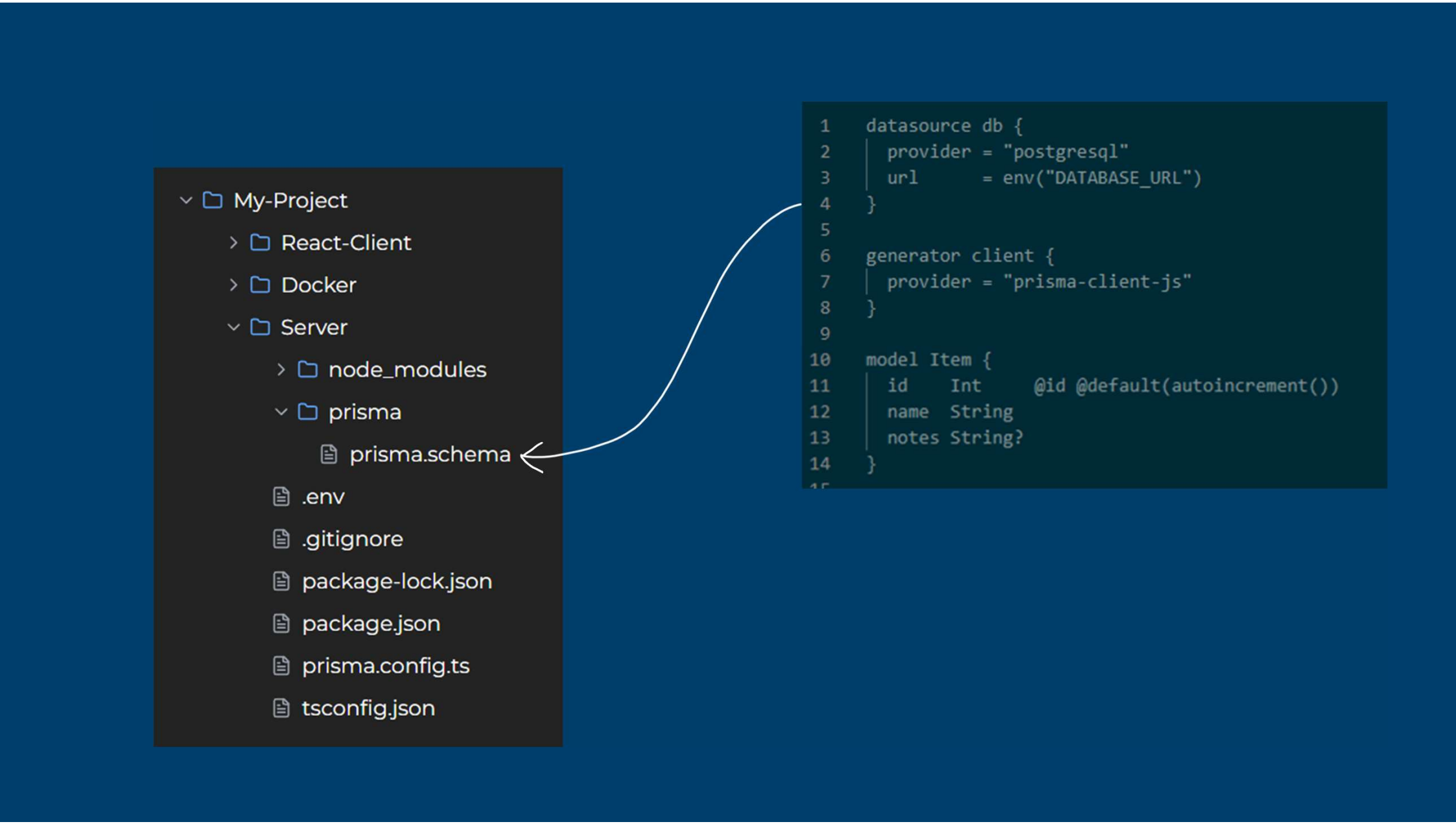
Don’t try this with Prisma 7.0.1

What is schema.prisma?

- Object-Relational Model configuration file
- Defines database tables with object types

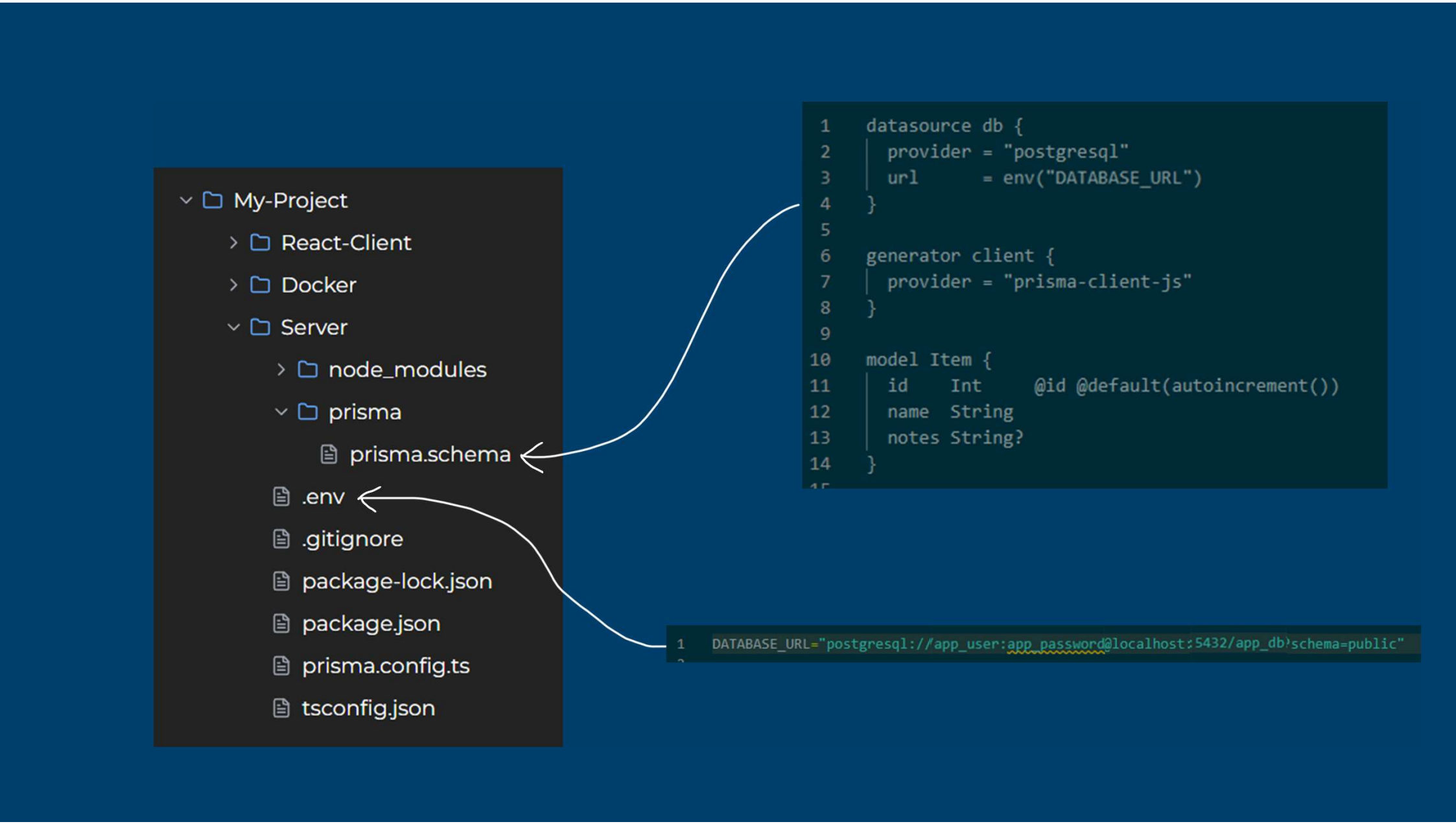
What is schema.prisma?

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}

model Item {
  id    Int     @id @default(autoincrement())
  name  String
  notes String?
}
```

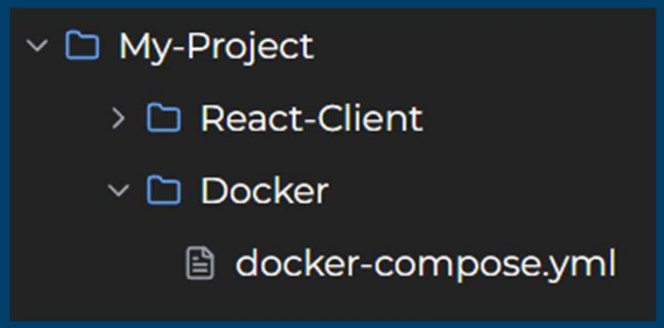

My-Project
React-Client
Docker
Server
node_modules
prisma
prisma.schema
.env
.gitignore
package-lock.json
package.json
prisma.config.ts
tsconfig.json
datasource db {
provider = "postgresql"
url = env("DATABASE_URL")
}
generator client {
provider = "prisma-client-js"
}
model Item {
id Int @id @default(autoincrement())
name String
notes String?
}

Step 3: Configure Prisma – ./React-Client/App.tsx

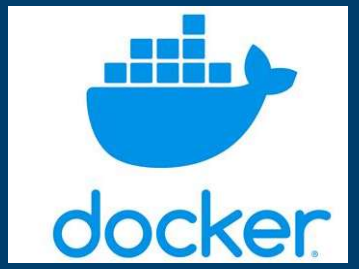
```tsx
import { useState } from "react";

interface Item {
  id: number;
  name: string;
}

function App() {
  const [items, setItems] = useState<Item[]>([]);
  const [name, setName] = useState("");
  const [search, setSearch] = useState("");

  const createItem = () => {
    if (!name.trim()) return;

    const newItem: Item = {
      id: items.length ? items.at(-1)!.id + 1 : 1,
      name,
    };
    setItems([...items, newItem]);
    setName("");
  };

  const searchItems = (q: string) => {
    setSearch(q);
  };

  const filteredItems = items.filter((item) =>
    item.name.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <div style={{ padding: "2rem", fontFamily: "sans-serif" }}>
      <h1>Items</h1>

      <input
        placeholder="New item"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <button onClick={createItem} style={{ marginLeft: "0.5rem" }}>
        Add
      </button>

      <hr style={{ margin: "1rem 0" }} />

      <input
        placeholder="Search"
        value={search}
        onChange={(e) => searchItems(e.target.value)}
      />

      <ul>
        {filteredItems.map((item) => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```


My-Project
React-Client
Docker
Server
node_modules
prisma
prisma.schema
.env
.gitignore
package-lock.json
package.json
prisma.config.ts
tsconfig.json
datasource db {
provider = "postgresql"
url = env("DATABASE_URL")
}
generator client {
provider = "prisma-client-js"
}
model Item {
id Int @id @default(autoincrement())
name String
notes String?
}
DATABASE_URL="postgresql://app_user:app_password@localhost:5432/app_db?schema=public"

./Docker/docker-compose.yml

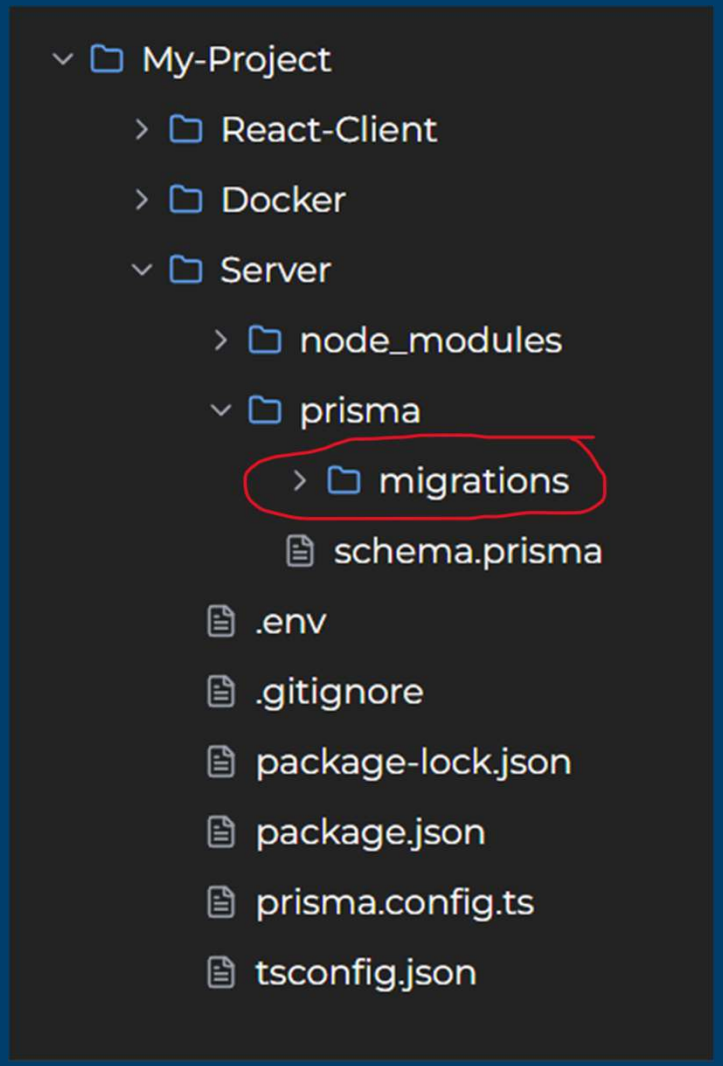



```yaml
version: '3.9'

Run All Services
services:
  Run Service
  db:
    image: postgres:16
    restart: always
    environment:
      POSTGRES_USER: app_user
      POSTGRES_PASSWORD: app_password
      POSTGRES_DB: app_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```



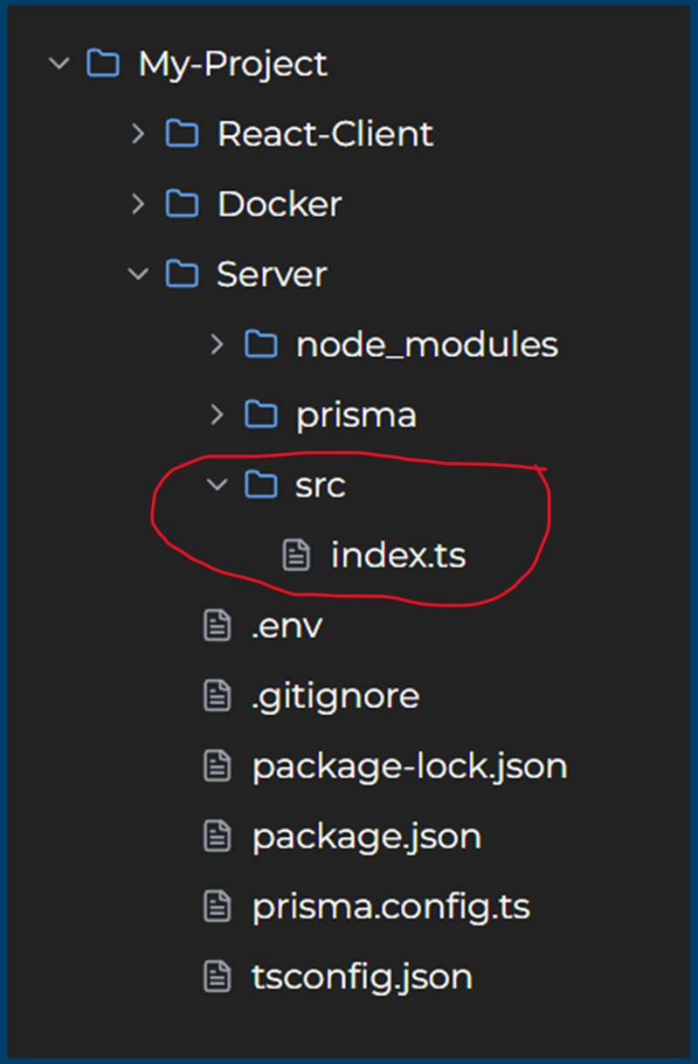
- Migrate schema.prisma

```
npx prisma migrate reset
npx prisma migrate dev --name init
npx prisma generate
```



Creating Express Web API

- Create ‘src/index.ts’



./Server/src/index.ts

```
1   import express from "express";
2   import cors from "cors";
3   import { PrismaClient } from "@prisma/client";
4
5   const prisma = new PrismaClient({
6     log: ["query", "info", "warn", "error"]
7   });
8
9   const app = express();
10
11  app.use(cors());
12  app.use(express.json());
```

./Server/src/index.ts

```
39  async function main() {
40    try {
41      await prisma.$connect();
42      console.log("Connected to database!");
43
44      app.listen(4000, () => {
45        console.log("Server running on http://localhost:4000");
46      });
47    } catch (err) {
48      console.error("❌ Server failed to start:");
49      console.error(err);
50      process.exit(1);
51    }
52  }
53
54  void main();
```

./Server/src/index.ts

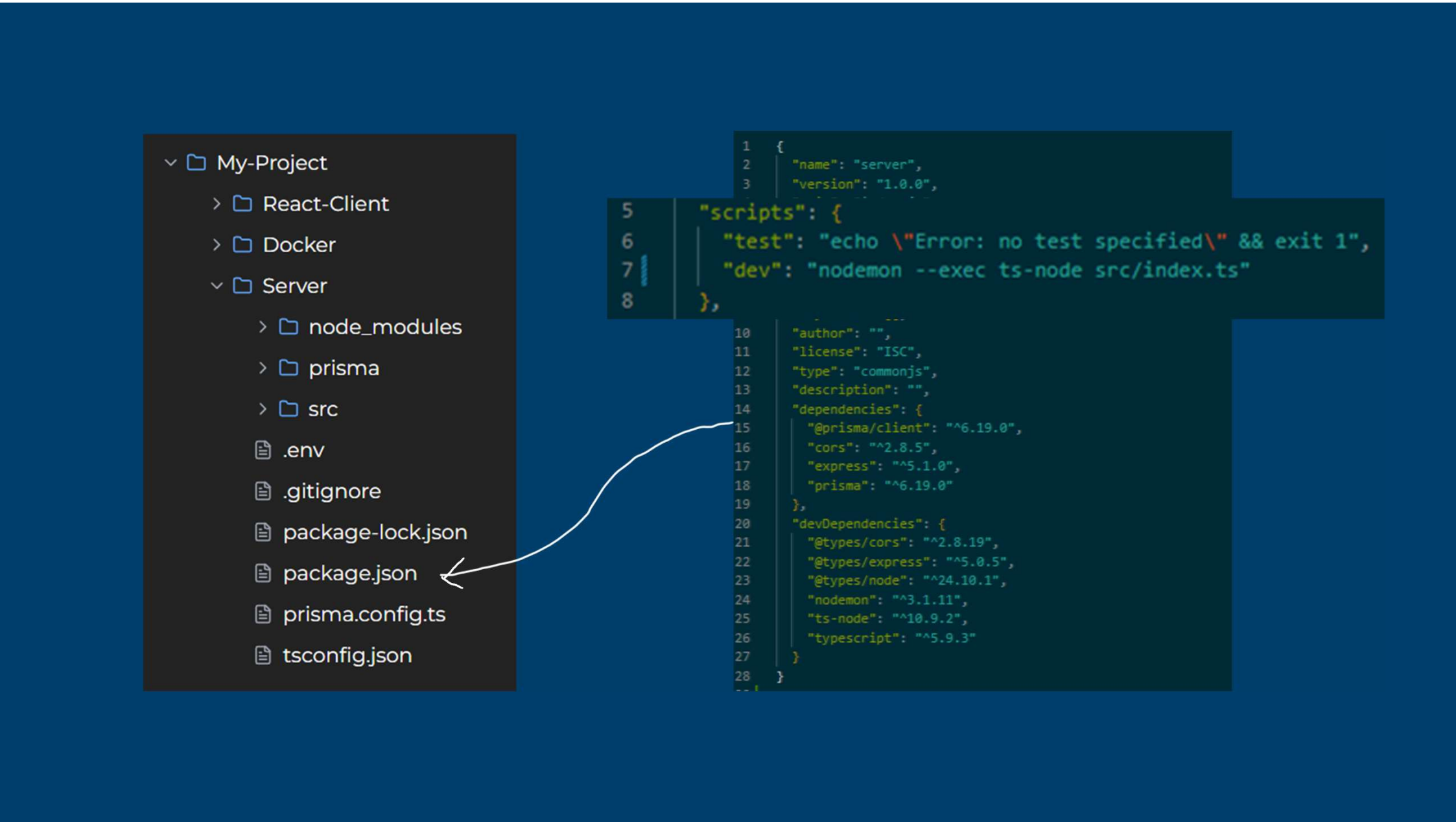
```ts
app.get("/items", async (req, res) => {
  const items = await prisma.item.findMany();
  res.json(items);
});

app.post("/items", async (req, res) => {
  const item = await prisma.item.create({ data: req.body });
  res.json(item);
});

app.get("/search", async (req, res) => {
  const q = typeof req.query.q === "string" ? req.query.q : "";

  const results = await prisma.item.findMany({
    where: {
      name: {
        contains: q,
        mode: "insensitive"
      }
    }
  });

  res.json(results);
});
```

GET
http://localhost:4000/items

POST
http://localhost:4000/items
Body: { “name” : “<ITEM NAME>” }

GET
http://localhost:4000/search
?q=<SEARCH>

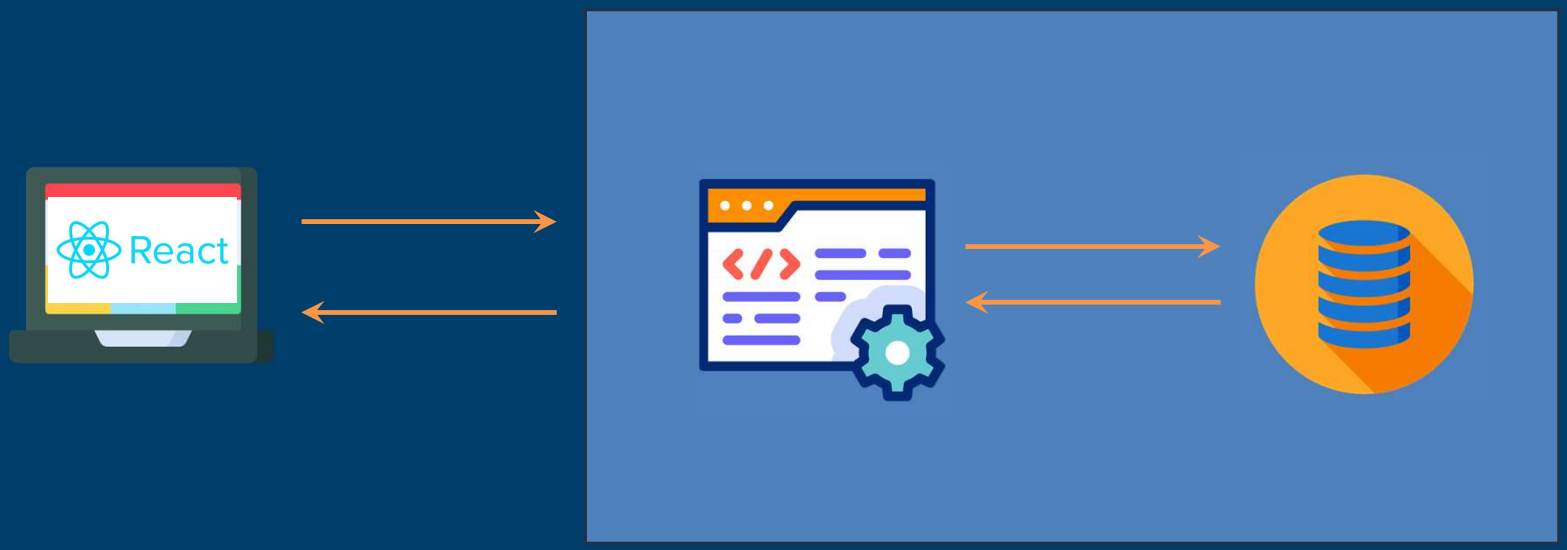
Running the Backend


My-Project
React-Client
Docker
Server
node_modules
prisma
src
.env
.gitignore
package-lock.json
package.json
prisma.config.ts
tsconfig.json
{
"name": "server",
"version": "1.0.0",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1",
"dev": "nodemon --exec ts-node src/index.ts"
},
"author": "",
"license": "ISC",
"type": "commonjs",
"description": "",
"dependencies": {
"@prisma/client": "^6.19.0",
"cors": "^2.8.5",
"express": "^5.1.0",
"prisma": "^6.19.0"
},
"devDependencies": {
"@types/cors": "^2.8.19",
"@types/express": "^5.0.5",
"@types/node": "^24.10.1",
"nodemon": "^3.1.11",
"ts-node": "^10.9.2",
"typescript": "^5.9.3"
}
}

./Server/

```
PS C:\Users\Callu\Documents\Tutorial\Postgres-Prisma-Tutorial\server> npm run dev

> server@1.0.0 dev
> nodemon --exec ts-node src/index.ts

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/index.ts`
prisma:info Starting a postgresql pool with 17 connections.
Connected to database!
Server running on http://localhost:4000
```

Connecting to the Backend



./React-Client/src/App.tsx

```tsx
import { useEffect, useState } from "react";

type Item = {
  id: number;
  name: string;
};

function App() {
  const [items, setItems] = useState<Item[]>([]);
  const [name, setName] = useState("");
  const [search, setSearch] = useState("");
```

./React-Client/src/App.tsx

```
12
13  const fetchItems = async (): Promise<void> => {
14    const res = await fetch("http://localhost:4000/items");
15    const data: Item[] = await res.json();
16    setItems(data);
17  };
18
```

GET
http://localhost:4000/items

./Server/src/index.ts

```
app.get("/items", async (req, res) => {
  const items = await prisma.item.findMany();
  res.json(items);
});
```

./React-Client/src/App.tsx

```
19  const createItem = async (): Promise<void> => {
20    await fetch("http://localhost:4000/items", {
21      method: "POST",
22      headers: { "Content-Type": "application/json" },
23      body: JSON.stringify({ name })
24    });
25    setName("");
26    fetchItems();
27  };
```

POST
http://localhost:4000/items
Body: { “name” : “<ITEM NAME>” }

Calls ./Server/src/index.ts

```
app.post("/items", async (req, res) => {
  const item = await prisma.item.create({ data: req.body });
  res.json(item);
});
```

./React-Client/src/App.tsx

```tsx
  const searchItems = async (q: string): Promise<void> => {
    setSearch(q);
    const res = await fetch("http://localhost:4000/search?q=" + q);
    const data: Item[] = await res.json();
    setItems(data);
  };
```

GET
http://localhost:4000/search
?q=<SEARCH>

Calls ./Server/src/index.ts

```ts
app.get("/search", async (req, res) => {
  const q = typeof req.query.q === "string" ? req.query.q : "";

  const results = await prisma.item.findMany({
    where: {
      name: {
        contains: q,
        mode: "insensitive"
      }
    }
  });

  res.json(results);
});
```

./React-Client/src/App.tsx

```tsx
  return (
    <div style={{ padding: "2rem", fontFamily: "sans-serif" }}>
      <h1>Items</h1>

      <input
        placeholder="New item"
        value={name}
        onChange={e => setName(e.target.value)}
      />
      <button onClick={createItem} style={{ marginLeft: "0.5rem" }}>
        Add
      </button>

      <hr style={{ margin: "1rem 0" }} />

      <input
        placeholder="Search"
        value={search}
        onChange={e => searchItems(e.target.value)}
      />

      <ul>
        {items.map((item: any) => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

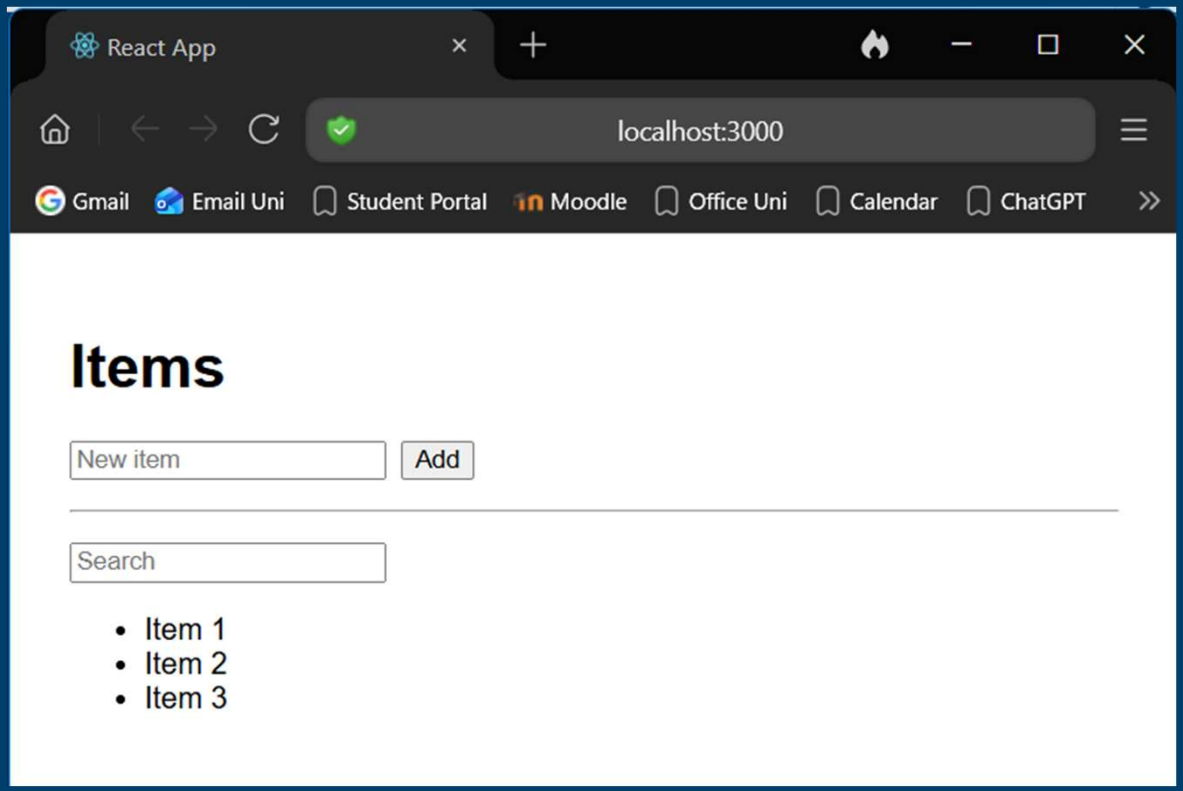
Finished!!

- ./Server/

```
npm run dev
```

- ./React-Client/

```
npm start
```



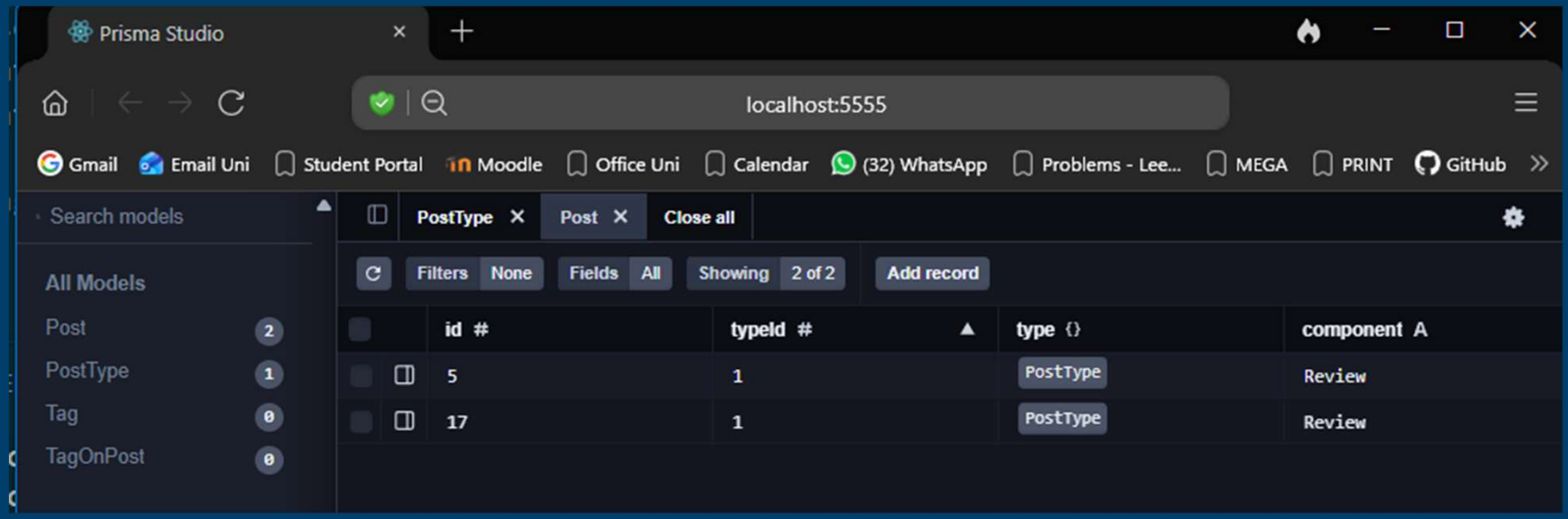
Advanced Tips

Security Considerations

- The demo is not suitable for prod
- Use cors to restrict request origin
- Authenticate users
- Validate Inputs
- Use HTTPS in production

Prisma Studio

```
\Postgres-Prisma-Tutorial\server> npx prisma studio
```



Scaling Databases

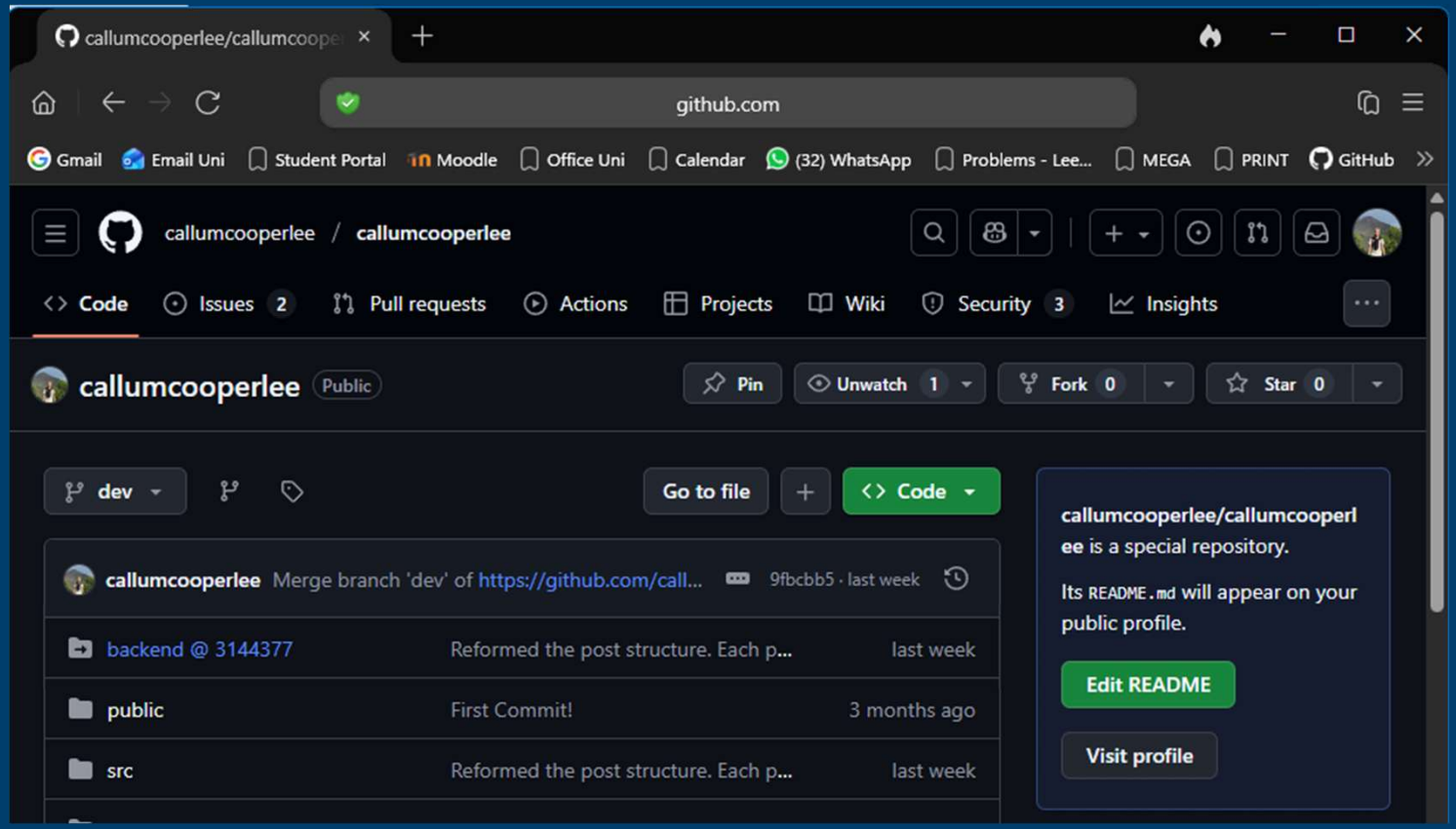
```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}

model PostType {
  id        Int      @id @default(autoincrement())
  name      String   @unique
  metadata  Json
  template  String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  posts     Post[]
}

model Post {
  id        Int      @id @default(autoincrement())
  typeId    Int
  type      PostType @relation(fields: [typeId], references: [id])
  component String
  metadata  Json
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  tags      TagOnPost[]
}

model Tag {
  id        Int        @id @default(autoincrement())
  name      String     @unique
  posts     TagOnPost[]
}

model TagOnPost {
  postId Int
  tagId  Int

  post   Post @relation(fields: [postId], references: [id])
  tag    Tag  @relation(fields: [tagId], references: [id])

  @@id([postId, tagId])
}
```

Remember to migrate using ‘npx prisma migrate’ every time shema.prisma is updated

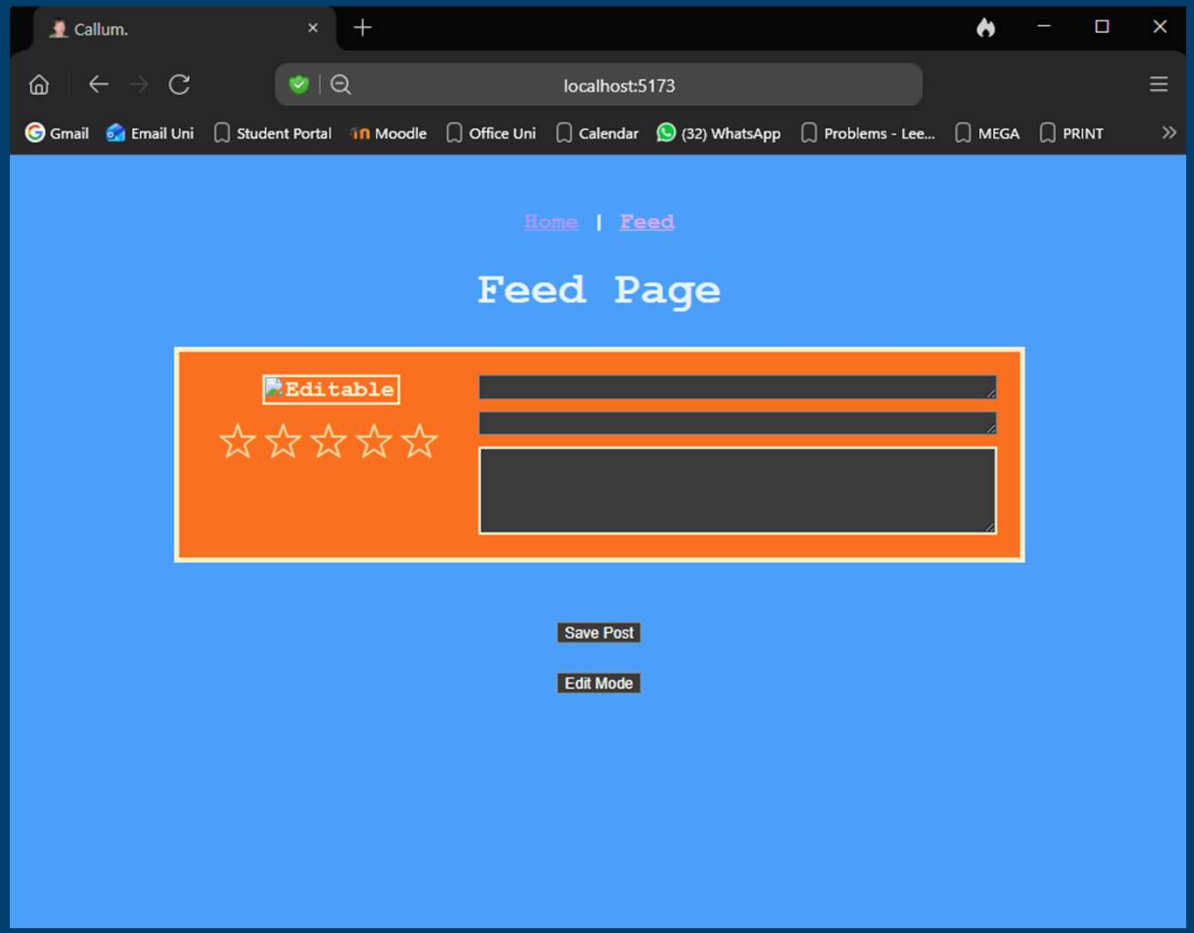
File Upload

```
45  router.post("/", upload.single("thumbnail"), async (req: Request, res: Response) => {
46    try {
47      const { typeId, metadata, component, tagIds } = req.body;
48      const parsedMetadata = metadata ? JSON.parse(metadata) : {};
49
50      if (req.file) {
51        parsedMetadata.thumbnail = `/uploads/${req.file.filename}`;
52      } else {
53        parsedMetadata.thumbnail = parsedMetadata.thumbnail || "/uploads/default.png";
54      }
55
56      const post = await prisma.post.create({
57        data: {
58          component,
59          metadata: parsedMetadata,
60          type: { connect: { id: Number(typeId) } },
61          tags: tagIds
62            ? { create: JSON.parse(tagIds).map((id: number) => ({ tagId: id })) }
63            : undefined,
64        },
65      });
66
67      res.status(201).json(post);
68    } catch (err) {
69      console.error(err);
70      res.status(500).json({ error: "Failed to create post" });
71    }
```

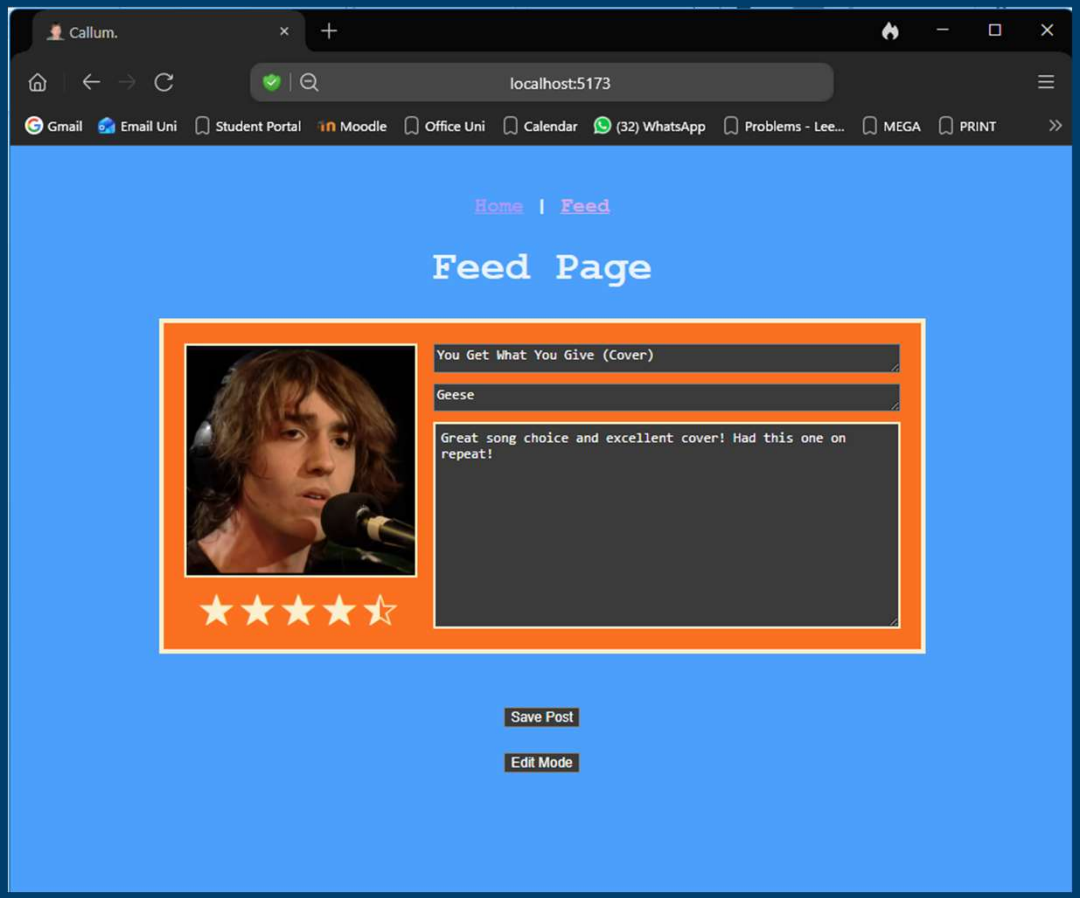
File Upload


callumcooperlee/callumcooper
github.com
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
(32) WhatsApp
Problems - Lee...
MEGA
PRINT
GitHub
callumcooperlee / callumcooperlee
Code
Issues 2
Pull requests
Actions
Projects
Wiki
Security 3
Insights
callumcooperlee Public
Pin
Unwatch 1
Fork 0
Star 0
dev
Go to file
Code
callumcooperlee Merge branch 'dev' of https://github.com/call... 9fbcbb5 · last week
backend @ 3144377 Reformed the post structure. Each p... last week
public First Commit! 3 months ago
src Reformed the post structure. Each p... last week
callumcooperlee/callumcooperlee is a special repository.
Its README.md will appear on your public profile.
Edit README
Visit profile

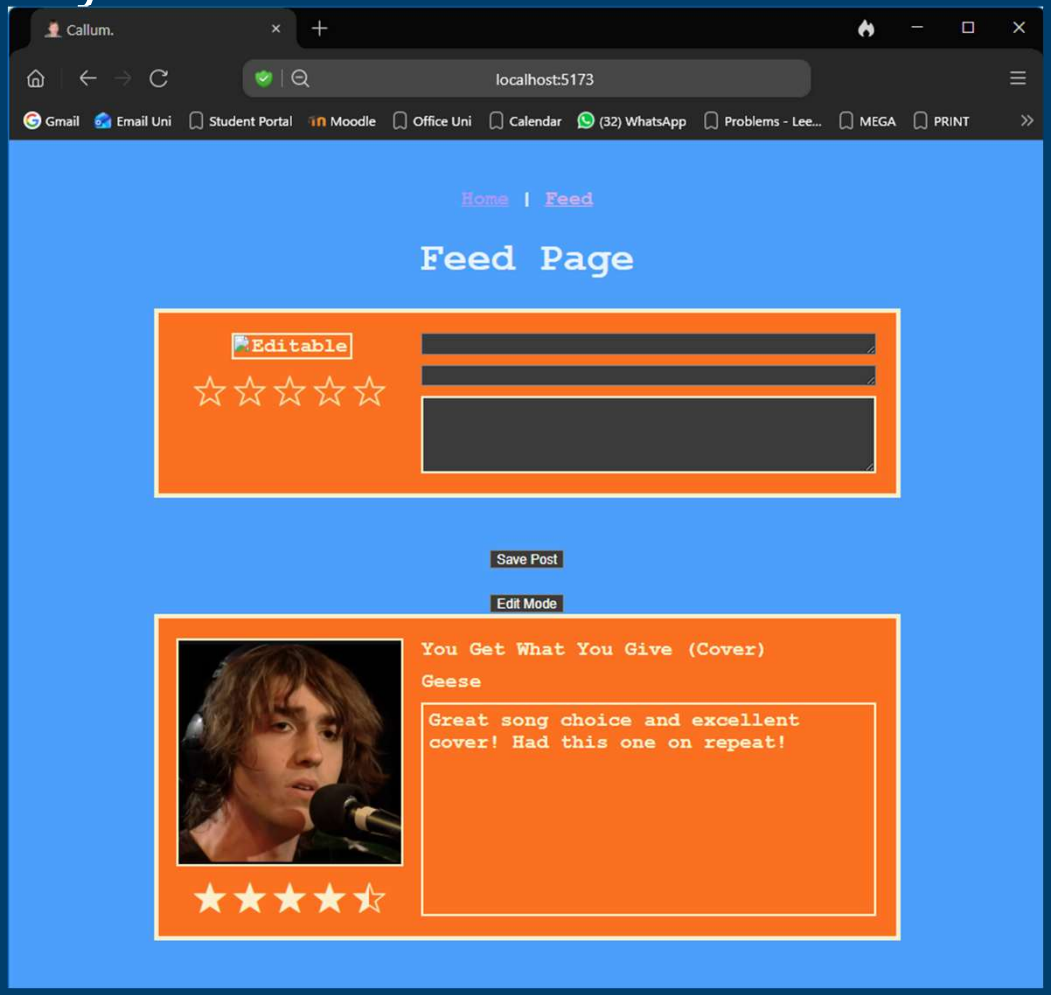
My website!


Callum.
localhost:5173
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
(32) WhatsApp
Problems - Lee...
MEGA
PRINT
Home | Feed
Feed Page
Editable
Save Post
Edit Mode

My website!


Callum.
localhost:5173
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
(32) WhatsApp
Problems - Lee...
MEGA
PRINT
Home | Feed
Feed Page
You Get What You Give (Cover)
Geese
Great song choice and excellent cover! Had this one on repeat!
Save Post
Edit Mode

My website!


Callum.
localhost:5173
Gmail
Email Uni
Student Portal
Moodle
Office Uni
Calendar
(32) WhatsApp
Problems - Lee...
MEGA
PRINT
Home | Feed
Feed Page
Editable
Save Post
Edit Mode
You Get What You Give (Cover)
Geese
Great song choice and excellent cover! Had this one on repeat!

Looking to the future..

- Easy to implement security considerations
- More complex table relationships
- Powerful and scalable queries
- Easy to move to a host device

Thanks For Listening! Any Questions?

Sources:

- https://create-react-app.dev/docs/adding-typescript/
- https://www.docker.com/products/docker-desktop/
- https://www.postgresql.org/about/
- https://en.wikipedia.org/wiki/ACID
- https://www.prisma.io/
- https://nodejs.org/en
- https://expressjs.com/
- https://expressjs.com/en/resources/middleware/cors.html
- https://medium.com/@byte.talking/multi-form-data-uploads-with-react-js-express-and-multer-b19adb3c1de2

Tutorial:

- https://github.com/callumcooperlee/Postgres-Prisma-Tutorial