spur-express

- spur-express github repo

Journey

- Setup Github Repo: spur-express github repo

- chore: bootstrap Express + TypeScript backend with health check and gitignore
- Project initialization

```
mkdir backend frontend
cd backend
npm init -y
npm install express cors dotenv
npm install -D typescript ts-node-dev @types/node @types/express
@types/cors
npx tsc --init
```

- tsconfig.json: Configured TypeScript for a stable Node.js backend with strict type safety and a clear source-to-build output flow.

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "commonjs",
    "rootDir": "src",
    "outDir": "dist",
    "strict": true,
    "esModuleInterop": true
  }
}
```

- src/app.ts: Created a minimal Express app with CORS and JSON middleware, plus a /health endpoint to verify server liveness.

```
import express from "express";
import cors from "cors";

const app = express();

app.use(cors());
app.use(express.json());
app.get("/health", (_, res) => {
  res.json({ status: "ok" });
});

export default app;
```

- src/server.ts: Separated server startup from app configuration and enabled environment variable loading for deployment readiness.

```
import app from "./app";
import dotenv from "dotenv";

dotenv.config();

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {
  console.log(`spur-express backend running on port ${PORT}`);
});
```

- package.json: Added development, build, and production scripts to support local development and deployment on Render.

```
"scripts": {
  "dev": "ts-node-dev --respawn src/server.ts",
  "build": "tsc",
  "start": "node dist/server.js"
}
```

- Started the backend in watch mode to ensure the server runs correctly during development.

```
npm run dev
```

- Health check validation

```
http://localhost:3000/health
```

- Confirmed the backend is running successfully and responding to requests.

```
{ "status": "ok" }
```

- feat: connect backend to Supabase Postgres and verify db health
- init supabase project 'spur-express'
  - db URI
    - conn. str. : `postgresql://postgres:<PASSWORD>@db.xxxxx.supabase.co:5432/postgres`
    - stored at `.env`

```
DATABASE_URL=postgresql://postgres:YOUR_PASSWORD@db.xxxxx.supabase.co:5432/
postgres
PORT=3000
```

- packs

```
npm install pg uuid
npm install -D @types/pg
```

- backend/src/db/index.ts: db client using pool, manager that keeps a small set of open database connections and reuses them instead of opening a new one for every query.

```
import { Pool } from "pg";

if (!process.env.DATABASE_URL) {
  throw new Error("DATABASE_URL is missing");
}

export const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false }, // REQUIRED for Supabase
});
```

- supabase sql query init:
  - conversations represents one chat session (thread) so all messages are grouped correctly.
  - messages stores each user/AI message with sender, order, and automatic cleanup via ON DELETE CASCADE.

```
CREATE TABLE IF NOT EXISTS conversations (
  id UUID PRIMARY KEY,
  created_at TIMESTAMP DEFAULT now()
);

CREATE TABLE IF NOT EXISTS messages (
  id UUID PRIMARY KEY,
  conversation_id UUID REFERENCES conversations(id) ON DELETE CASCADE,
  sender TEXT CHECK (sender IN ('user', 'ai')),
  text TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT now()
);
```

- Modify src/app.ts: Verify DB connection from Express

```
import express from "express";
import cors from "cors";
import { pool } from "./db";

const app = express();

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

app.get("/health", (_, res) => {
  res.json({ status: "ok" });
});
app.get("/db-health", async (_, res) => {const result = await
pool.query("SELECT now()");
  res.json({ db: "connected", time: result.rows[0] });
});

export default app;
```

- Modify src/server.ts: Move dotenv.config() to the TOP, Node evaluates imports first, so dotenv.config() must run before importing anything

```
import dotenv from "dotenv"; // Node evaluates imports first, so
dotenv.config() must run before importing anything that reads process.env.

dotenv.config();

import app from "./app";

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {
  console.log(`spur-express backend running on port ${PORT}`);
});
```

- run & check

```
npm run dev
```

```
http://localhost:3000/db-health
```

```
{"db":"connected","time":{"now":"2025-12-18T22:20:54.578Z"}}
```