🚀 Full Stack Blog App Documentation (Node.js + Express + EJS + PostgreSQL + Prisma)

📌 Overview

This is a full-stack **blog and user-auth app** using:

- **Node.js + Express** for backend logic
- **EJS** for templating UI
- **Prisma ORM** for database operations with **PostgreSQL**
- **Session-based Authentication** with CSRF, rate limiting, Helmet for security

---

⚙️ Installation & Setup

🧰 1. Prerequisites

Install the following:

- Node.js – JavaScript runtime
- PostgreSQL – Relational database
- Prisma CLI – ORM and DB toolkit

```
npm install -g prisma
```

📥 2. Clone Project

```
git clone <your_repo_url>
cd blogexpressapp
```

📦 3. Install Dependencies

```
npm install
```

From `package.json`, the main packages:

Package	Purpose
`express`	Core backend framework
`ejs`	Server-side view rendering
`prisma`, `@prisma/client`	ORM for PostgreSQL
`bcryptjs`	Password hashing
`express-session`	Auth session management
`body-parser`, `cookie-parser`	Parsing request data
`csurf`	CSRF protection middleware
`helmet`	Adds security headers
`express-rate-limit`	Prevents brute force attacks
`method-override`	Supports PUT/DELETE in forms

Package	Purpose
`express-ejs-layouts`	Template layout structure

🛠 4. Setup PostgreSQL & Prisma

1. Create a PostgreSQL DB
2. Configure `.env`:

```
DATABASE_URL="postgresql://user:password@localhost:5432/yourdbname"
SESSION_SECRET="yoursecret"
```

3. Initialize Prisma:

```
npx prisma init
```

4. Define your schema in `prisma/schema.prisma`:

```
model User {
  id        Int      @id @default(autoincrement())
  email     String   @unique
  username  String
  password  String
  createdAt DateTime @default(now())
}
```

5. Migrate your DB:

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

📁 Project Structure

```
project/
│
├── routes/
│   └── auth_routes.js
│
├── views/
│   ├── home.ejs
│   ├── login.ejs
│   ├── register.ejs
│   ├── dashboard.ejs
│   ├── pin.ejs
│   └── error.ejs
│
├── blogs.js
├── server.js
├── package.json
├── .env
├── prisma/
│   └── schema.prisma
```

📄 File-by-File Breakdown

`server.js`

- Sets up Express server
- Uses middleware:
    - Body parser
    - Session

    - CSRF protection
    - Helmet
- Loads routes from `auth_routes.js`
- Sets up EJS as the templating engine
- Error handler for undefined routes

`auth_routes.js`

Handles routes related to:

- `/register` – GET/POST registration
- `/login` – GET/POST login
- `/dashboard` – Protected dashboard
- `/logout` – Destroy session

 Middleware:

- Checks if session exists
- Uses bcrypt to hash/verify passwords
- Stores session on login

 Enhancement: Should connect with Prisma to create and query users in the DB.

`blogs.js`

- Static array of blog posts with `{ id, title, content }`
- Used by `pin.ejs` to render sample blog content

 Future: Replace with Prisma blog model and fetch dynamically

---

EJS Templates (`views/`)

File	Purpose
`home.ejs`	Public homepage. Shows links to login/register.
`login.ejs`	Login form UI with email/password input
`register.ejs`	Registration form with validation, CAPTCHA
`dashboard.ejs`	Shows user's info post-login. Welcoming layout
`pin.ejs`	Shows full blog post details
`error.ejs`	Renders custom error message

All templates use Bootstrap and simple card-based layouts.

`package.json`

```
{
  "name": "blogexpressapp",
  "main": "index.js",
  "type": "commonjs",
  "dependencies": {
    "@prisma/client": "^6.5.0",
    "bcryptjs": "^3.0.2",
    ...
  }
}
```

Note: `index.js` is set as the main entry but the real entry is `server.js`. Update if needed.

Security Highlights

- `helmet` protects against XSS, clickjacking, etc.
- `express-rate-limit` defends against brute-force login
- `csurf` adds CSRF protection on form requests
- `bcryptjs` securely hashes passwords

Running the App

```
npm run dev      # OR npm start
```

Go to `http://localhost:3000`