Backend Setup Instructions

REST API using Node.js, Express, and MongoDB

---

1. Prerequisites

- Node.js (v20.x recommended)
- npm (comes with Node.js)
- MongoDB (Local installation or MongoDB Atlas)
- Git

2. Clone the Repository

```
git clone <your-github-repo-url>
cd <repo-folder-name>
```

3. Install Dependencies

```
npm install
```

- express
- mongoose
- cors
- cookie-parser
- dotenv
- bcrypt
- jsonwebtoken
- nodemon

4. Configure Environment Variables

Create a .env file in the project root and add:

```
PORT=9000
MONGO=mongodb://localhost:27017/productStore
SECRET_KEY=your_secret_key_here
```

5. Start MongoDB

For local MongoDB:

```
mongod
```

For MongoDB Atlas: Replace the MONGO value in .env with your Atlas connection string.

6. Run the Backend Server

```
npm start
```

Expected console output:

```
Connected to MongoDB
Server running on port 9000
```

7. API Endpoints

```
User Routes (/user)
POST   /user/user-create
GET    /user/get-data
GET    /user/get-data-id
PUT    /user/update-data-id/:id
DELETE /user/delete-data-id/:id

Auth Routes (/auth)
POST   /auth/user-login
POST   /auth/user-logout

Product Routes (/product)
POST   /product/create
GET    /product/get
GET    /product/get/:id
PUT    /product/update/:id
DELETE /product/delete/:id
```

8. CORS & Cookies Configuration

```
app.use(cors({ origin: "http://localhost:5173", credentials: true }));
```

Ensure the origin matches your frontend URL. JWT authentication works because credentials: true is enabled.

9. Debugging Tips

- 500 Internal Server Error: Check MongoDB connection and environment variables.
- 404 Not Found: Verify frontend API URLs match backend routes.
- Cross-Origin Issues: Confirm correct CORS origin and credentials settings.
- Use console.log in route handlers for debugging.