Create login page in Docker

I want to create an html login page and deploy it on docker in simple and also want to store the data in mongo in simple give me steps

Alright, let's keep this **simple, straight**, and doable even if you're in a hurry.
Here's the clean roadmap to create an **HTML login page**, **store data in MongoDB**, and **deploy everything in Docker**.

✅ Step-by-Step Guide (Simple Version)

1️⃣ Create a Simple Login Page (login.html)

Make a folder **`project/`**
Inside it create **`login.html`**:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="/login" method="POST">
        <input type="text" name="username" placeholder="Enter username" required><br><br>
        <input type="password" name="password" placeholder="Enter password" required><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

2️⃣ Create a Simple Backend Using Node.js + Express

Inside **`project/`**, run:

```bash
npm init -y
npm install express mongoose body-parser cors
```

Create a file **`server.js`**:

```js
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname)); // to serve login.html

mongoose.connect("mongodb://mongo:27017/loginDB")
  .then(() => console.log("MongoDB connected"))
  .catch(err => console.log(err));

const userSchema = new mongoose.Schema({
  username: String,
  password: String
});
const User = mongoose.model("User", userSchema);

app.post("/login", async (req, res) => {
  const user = new User({
    username: req.body.username,
    password: req.body.password,
  });
  await user.save();
  res.send("Login Data Saved Successfully!");
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

3️⃣ Create Dockerfile for Node App

Inside **`project/`**, create **`Dockerfile`**:

```Dockerfile
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

4️⃣ Create Docker Compose (to run Node + MongoDB)

In the same folder **`project/`**, create **`docker-compose.yml`**:

```yaml
version: "3"
services:
  app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - mongo

  mongo:
    image: mongo
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

5️⃣ Run Everything with ONE Command

Inside your project folder:

```bash
docker-compose up --build
```

This will:

✔️ Build your Node backend
✔️ Run MongoDB container
✔️ Open the app on http://localhost:3000/login.html↗
✔️ Store submitted data in MongoDB

🎉 Your App Flow

1. Open browser → http://localhost:3000/login.html↗
2. Submit username + password
3. Node receives it and stores in MongoDB
4. Data gets saved in `loginDB` inside the Mongo container

If you want, I can also give:

✅ Folder structure
✅ A better styled login page
✅ Dashboard page after login
✔️ You tell me.

can u give me porject structure

Sure — here's the **clean and simple project structure** you should use for this HTML + Node.js + MongoDB + Docker setup.

📂 Project Structure (Simple & Correct for Docker + Node + MongoDB)

```pgsql
project/
│
├── login.html
├── server.js
├── package.json
├── package-lock.json   (auto created)
│
├── Dockerfile
├── docker-compose.yml
│
└── node_modules/       (auto created after npm install)
```

📘 Explanation of Each File

1. login.html

Your simple login form (frontend).

2. server.js

Node.js Express server handling:

- serving login.html
- saving username & password to MongoDB

3. package.json

Stores project dependencies (express, mongoose, etc.).

4. Dockerfile

Creates a Docker image for your Node.js app.

5. docker-compose.yml

Runs both:

- Node app container
- MongoDB container
  together with one command.

6. node_modules/

Created automatically after `npm install`, not edited manually.

🏁 Final View

```pgsql
project/
│
├── login.html
├── server.js
│
├── package.json
├── package-lock.json
│
├── Dockerfile
├── docker-compose.yml
│
└── node_modules/
```

If you want, I can also give you a **downloadable ZIP** of this full project or help you **run it step-by-step**.

yes

Alright — before I generate the **downloadable ZIP**, just tell me one thing so I package it correctly:

Which version do you want?

Option A — Very Simple Project

✔️ Only HTML login page
✔️ Node.js backend
✔️ Stores username + password in MongoDB
✔️ Shows “Saved Successfully!” after login
✔️ Minimal code

Option B — Slightly Better Project

✔️ HTML login
✔️ Backend validation
✔️ Redirects to a dashboard page
✔️ Proper folder separation
✔️ Cleaner UI

Option C — Full Beginner-Friendly Project

✔️ Login + Signup
✔️ Password hashing
✔️ Flash messages
✔️ Dashboard
✔️ Proper folder structure
✔️ Clean UI
✔️ Ready for deployment

Which one should I package into the ZIP?

A, B, or C?