Updating the Web Blog Application

READY FOR DATABASE CONNECTION + ADMIN CREATION

PART 1 — Updating All Frontend Components

Objective:

Students will update each React component so the blog works with user login, admin dashboard, blog posts, and comments.

1. Update the following components

   - `AppNavbar.js`
   - `HomePage.js`
   - `About.js`
   - `Services.js`
   - `Contact.js`
   - `Login.js`
   - `Register.js`
   - `Dashboard.js`
   - `AdminPost.js`
   - `BlogPost.js`
   - `LogoutModal.js`
   - `Footer.js`

About.js

```
export default function About() {
 return (
  <div className="container mt-4">
   <h2>About</h2>
   <p>Simple blog application built with React and Express.</p>
  </div>
 );
}
```

AdminPost.js (New File)

```
export default function AdminPost() {
 const user = JSON.parse(sessionStorage.getItem("user"));

 if (!user || user.role !== "admin") {
  return (
```

```
    <div className="container mt-4">
      <h2>Access Denied</h2>
      <p>Administrators only.</p>
    </div>
  );
 }

 return (
  <div className="container mt-4">
    <h2>Manage Posts</h2>
    <p>Future admin tools to edit/delete posts.</p>
  </div>
 );
}
```

AppNavbar.js

```
import { useState } from "react";
import { Link } from "react-router-dom";
import LogoutModal from "./LogoutModal.js";

export default function AppNavbar({ currentUser, setCurrentUser }) {
 const [showLogout, setShowLogout] = useState(false);

 // Logout handler
 const handleLogout = () => {
  sessionStorage.removeItem("user"); // remove session
  setCurrentUser(null); // update navbar immediately
  setShowLogout(false); // close modal
 };

 return (
  <>
   <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
    <div className="container-fluid">
     <Link className="navbar-brand" to="/">My Blog</Link>

     <button
      className="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
```

```
        data-bs-target="#mainNavbar"
        aria-controls="mainNavbar"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span className="navbar-toggler-icon"></span>
      </button>

      <div className="collapse navbar-collapse" id="mainNavbar">
        <ul className="navbar-nav ms-auto">
          {/* Guest Links */}
          {!currentUser && (
            <>
              <li className="nav-item"><Link className="nav-link"
to="/about">About</Link></li>
              <li className="nav-item"><Link className="nav-link"
to="/services">Services</Link></li>
              <li className="nav-item"><Link className="nav-link"
to="/contact">Contact</Link></li>
              <li className="nav-item"><Link className="nav-link"
to="/login">Login</Link></li>
              <li className="nav-item"><Link className="nav-link"
to="/register">Register</Link></li>
            </>
          )}

          {/* Regular User Links */}
          {currentUser?.role === "user" && (
            <>
              <li className="nav-item"><Link className="nav-link"
to="/home">Home</Link></li>
              <li className="nav-item">
                <button className="nav-link btn btn-link" onClick={() =>
setShowLogout(true)}>Logout</button>
              </li>
            </>
          )}

          {/* Admin Links */}
          {currentUser?.role === "admin" && (
            <>
```

```
            <li className="nav-item"><Link className="nav-link"
to="/dashboard">Dashboard</Link></li>
            <li className="nav-item"><Link className="nav-link"
to="/admin/post">Manage Posts</Link></li>
            <li className="nav-item">
              <button className="nav-link btn btn-link" onClick={() =>
setShowLogout(true)}>Logout</button>
            </li>
           </>
          )}
         </ul>
        </div>
      </div>
    </nav>

    {/* Logout Confirmation Modal */}
    <LogoutModal
      show={showLogout}
      onClose={() => setShowLogout(false)}
      onConfirm={handleLogout}
    />
   </>
  );
}
```

BlogPost.js (New File)

```
// BlogPost.js
import { useParams } from "react-router-dom";
import { useEffect, useState } from "react";

export default function BlogPost() {
  const { id } = useParams();
  const [post, setPost] = useState(null);
  const [comments, setComments] = useState([]);
  const [text, setText] = useState("");

  useEffect(() => {
    // ⬅️ NOTE: Post fetching URL is likely incorrect too. Assuming /api/blog is correct for
your routes
```

```
  // Correct URL for fetching a single post from blogRoutes.js
  fetch(`http://localhost:5000/api/blog/${id}`)
    .then(res => res.json())
    .then(data => setPost(data));

  // ⬅️ FIX: Updated URL to use the new /api/comments endpoint
  fetch(`http://localhost:5000/api/comments/${id}`)
    .then(res => res.json())
    .then(data => setComments(data));
}, [id]);

function addComment() {
  // ⬅️ FIX: Updated URL to use the new /api/comments endpoint
  fetch("http://localhost:5000/api/comments", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      postId: id,
      user: "testUser", // TEMPORARY no authentication for testing
      text: text
    })
  })
    .then(res => res.json())
    .then(() => {
      // Fetch comments again instead of a full page reload for better user experience
      fetch(`http://localhost:5000/api/comments/${id}`)
        .then(res => res.json())
        .then(data => setComments(data));
      setText(""); // Clear the input field
    })
    .catch(err => console.error("Error posting comment:", err));
}

if (!post) return <p>Loading...</p>;

return (
  <div className="container mt-4">
    <h2>{post.title}</h2>
    <p>{post.content}</p>

    <hr />
```

```
    <h4>Comments</h4>
    <ul className="list-group mb-3">
     {comments.map((c, index) => (
      <li key={c._id || index} className="list-group-item"> {/* Use c._id for key if
available */}
        <strong>{c.user}:</strong> {c.text}
      </li>
     ))}
    </ul>

    <textarea
     className="form-control"
     placeholder="Write a comment..."
     value={text} // Bind value for controlled component
     onChange={(e) => setText(e.target.value)}
    ></textarea>

    <button className="btn btn-primary mt-2" onClick={addComment}
disabled={!text.trim()}>
     Post Comment
    </button>
   </div>
  );
}
```

Contact.js

```
export default function Contact() {
 return (
  <div className="container mt-4">
   <h2>Contact Me</h2>
   <p>You can reach me at:</p>

   <ul>
    <li>Email: example@myblog.com</li>
    <li>Facebook: @myblogpage</li>
    <li>Github: github.com/myblog</li>
   </ul>
  </div>
 );
}
```

Dashboard.js

```
import React from "react";

export default function Dashboard() {
 const user = JSON.parse(sessionStorage.getItem("user"));

 if (!user || user.role !== "admin") {
  return (
   <div className="container mt-4">
    <h2>Access Denied</h2>
    <p>This area is restricted to administrators.</p>
   </div>
  );
 }

 return (
  <div className="container mt-4">
   <h2>Admin Dashboard</h2>
   <p>Welcome back, <strong>{user.username}</strong>!</p>
   <p>Here you can manage posts and users (future feature).</p>
  </div>
 );
}
```

Footer.js

```
export default function Footer() {
 return (
  <footer className="bg-dark text-white text-center p-3 mt-auto">
   <p className="mb-0">&copy; 2025 MyBlog — All Rights Reserved</p>
  </footer>
 );
}
```

HomePage.js

```
import React from "react";

export default function HomePage() {
 return (
  <div className="container mt-5">
   {/* Hero Section */}
   <div className="text-center mb-5">
    <h1 className="display-4 fw-bold">Welcome to My Blog</h1>
    <p className="lead text-muted">
     Discover insights, stories, and tutorials crafted to inspire and inform.
    </p>
    <button className="btn btn-primary btn-lg mt-3">Explore Posts</button>
   </div>

   <hr className="mb-5" />

   {/* Category Cards */}
   <div className="row text-center">
    <div className="col-md-4 mb-4">
     <div className="card shadow-sm h-100">
      <div className="card-body">
       <i className="bi bi-journal-text display-4 text-primary mb-3"></i>
       <h4 className="card-title">Latest Posts</h4>
       <p className="card-text">
        Stay updated with the newest articles and trending topics.
       </p>
       <a href="#latest" className="btn btn-outline-primary btn-sm">
        Read More
       </a>
      </div>
     </div>
    </div>

    <div className="col-md-4 mb-4">
     <div className="card shadow-sm h-100">
      <div className="card-body">
       <i className="bi bi-cpu display-4 text-success mb-3"></i>
       <h4 className="card-title">Technology</h4>
       <p className="card-text">
```

```
            Explore programming tutorials, coding tips, and tech innovations.
           </p>
           <a href="#tech" className="btn btn-outline-success btn-sm">
            Read More
           </a>
          </div>
         </div>
        </div>

        <div className="col-md-4 mb-4">
         <div className="card shadow-sm h-100">
          <div className="card-body">
           <i className="bi bi-people display-4 text-warning mb-3"></i>
           <h4 className="card-title">Personal Stories</h4>
           <p className="card-text">
            Dive into life lessons, experiences, and meaningful reflections.
           </p>
           <a href="#stories" className="btn btn-outline-warning btn-sm">
            Read More
           </a>
          </div>
         </div>
        </div>
       </div>

       {/* Footer Call-to-Action */}
       <div className="text-center mt-5">
        <h5 className="fw-light">Want to stay updated?</h5>
        <button className="btn btn-dark mt-2">Subscribe to Newsletter</button>
       </div>
      </div>
    );
}
```

Login.js

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

const Login = ({ onLogin }) => {
  const [email, setEmail] = useState('');
```

```
const [password, setPassword] = useState('');
const [message, setMessage] = useState('');
const [loading, setLoading] = useState(false);
const navigate = useNavigate();

const API_URL = 'http://localhost:5000/api/auth/login';

const handleLogin = async (e) => {
 e.preventDefault();
 setMessage('');
 setLoading(true);

 try {
  const response = await fetch(API_URL, {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify({ email: email.trim(), password: password.trim() }),
  });

  const data = await response.json();

  if (response.ok) {
   const userData = data.user;
   sessionStorage.setItem('user', JSON.stringify(userData));

   if (onLogin) onLogin(userData); // updates navbar immediately

   // Redirect based on role
   if (userData.role === 'admin') navigate('/dashboard');
   else navigate('/home');
  } else {
   setMessage(data.message || 'Login failed.');
  }
 } catch (err) {
  console.error(err);
  setMessage('Cannot connect to server.');
 } finally {
  setLoading(false);
 }
};
```

```
  const alertClass = message.includes('failed') || message.includes('Cannot') ? 'alert-
danger' : 'alert-success';

  return (
    <div className="d-flex justify-content-center align-items-center vh-100 bg-light">
      <div className="card p-4 shadow-lg" style={{ maxWidth: '400px', width: '100%' }}>
        <h2 className="h3 text-center text-success mb-4 fw-bold">User Login</h2>

      {message && <div className={` alert ${alertClass} rounded `}
role="alert">{message}</div>}

        <form onSubmit={handleLogin}>
          <div className="mb-3">
            <label className="form-label fw-bold">Email</label>
            <input
              type="email"
              className="form-control"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              required
            />
          </div>
          <div className="mb-3">
            <label className="form-label fw-bold">Password</label>
            <input
              type="password"
              className="form-control"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              required
            />
          </div>
          <div className="d-grid mt-4">
            <button type="submit" className="btn btn-success fw-bold"
disabled={loading}>
              {loading ? 'Logging in...' : 'Log In'}
            </button>
          </div>
        </form>
      </div>
    </div>
```

```
  );
};

export default Login;
```

LogoutModal.js

```
import React from "react";

export default function LogoutModal({ show, onClose, onConfirm }) {
  if (!show) return null;

  return (
    <div className="modal fade show d-block" style={{ backgroundColor:
"rgba(0,0,0,0.5)" }}>
      <div className="modal-dialog modal-dialog-centered">
        <div className="modal-content">
          <div className="modal-header">
            <h5 className="modal-title">Confirm Logout</h5>
            <button type="button" className="btn-close" onClick={onClose}></button>
          </div>
          <div className="modal-body">
            <p>Are you sure you want to logout?</p>
          </div>
          <div className="modal-footer">
            <button className="btn btn-secondary" onClick={onClose}>Cancel</button>
            <button className="btn btn-danger" onClick={onConfirm}>Logout</button>
          </div>
        </div>
      </div>
    </div>
  );
}
```

Regsiter.js

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function Register() {
 const [form, setForm] = useState({ username: "", email: "", password: "", role: "user" });
 const [message, setMessage] = useState("");
 const navigate = useNavigate();

 const API_URL = "http://localhost:5000/api/auth/register";

 const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value });

 const handleSubmit = async (e) => {
  e.preventDefault();
  setMessage("");
  try {
   const res = await fetch(API_URL, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(form),
   });
   const data = await res.json();
   if (res.ok) {
    setMessage(data.message || "Registered successfully!");
    setTimeout(() => navigate("/login"), 1000);
   } else {
    setMessage(data.message || "Registration failed.");
   }
  } catch (err) {
   console.error(err);
   setMessage("Server error.");
  }
 };

 const alertClass = message.includes("success") ? "alert-success" : "alert-danger";

 return (
  <div className="d-flex justify-content-center align-items-center vh-100 bg-light">
```

```
    <div className="card p-4 shadow-lg" style={{ maxWidth: "450px", width: "100%" }}>
      <h2 className="h3 text-center text-primary mb-4 fw-bold">Register</h2>

      {message && <div className={` alert ${alertClass} rounded `}
role="alert">{message}</div>}

      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <label className="form-label fw-bold">Username</label>
          <input
            name="username"
            type="text"
            className="form-control"
            value={form.username}
            onChange={handleChange}
            required
          />
        </div>
        <div className="mb-3">
          <label className="form-label fw-bold">Email</label>
          <input
            name="email"
            type="email"
            className="form-control"
            value={form.email}
            onChange={handleChange}
            required
          />
        </div>
        <div className="mb-3">
          <label className="form-label fw-bold">Password</label>
          <input
            name="password"
            type="password"
            className="form-control"
            value={form.password}
            onChange={handleChange}
            required
          />
        </div>
```

```
        {/* Optional: allow selecting admin/user during testing */}
        <div className="mb-3">
          <label className="form-label fw-bold">Role</label>
          <select name="role" className="form-select" value={form.role}
onChange={handleChange}>
            <option value="user">User</option>
            <option value="admin">Admin</option>
          </select>
        </div>

        <div className="d-grid">
          <button type="submit" className="btn btn-primary fw-bold">Register</button>
        </div>
      </form>
      <p className="mt-3 text-center">
        Already have an account? <span className="text-primary text-decoration-
underline" style={{ cursor: "pointer" }} onClick={() => navigate("/login")}>Login</span>
      </p>
    </div>
   </div>
  );
}
```

Services.js

```
import { Link } from 'react-router-dom';

export default function Services() {
  return (
    <div className="container mt-4">
      <h2>Blog Articles</h2>
      <p className="text-muted">Here are some sample blog entries:</p>

      <div className="list-group">
        <Link to="/articles/1" className="list-group-item list-group-item-action">
          🚀 Blog Article #1
        </Link>
        <Link to="/articles/2" className="list-group-item list-group-item-action">
          💡 Blog Article #2
        </Link>
```

```
          <Link to="/articles/3" className="list-group-item list-group-item-action">
            📘 Blog Article #3
          </Link>
        </div>
      </div>
    );
  }
```

3. Copy the backend files exactly
    - server.js
    - authRoutes.js
    - blogRoutes.js
    - commentRoutes.js
    - userModel.js
    - blogPost.js
    - commentModel.js

authRoutes.js

```
import express from "express";
import User from "./userModel.js"; // ⬅️ FIX: Added .js extension

const router = express.Router();

// ===== REGISTER =====
router.post("/register", async (req, res) => {
 try {
  const { username, email, password, role } = req.body;
  // ... (validation logic remains the same) ...

  const normalizedEmail = email.trim().toLowerCase();
  const trimmedPassword = password.trim();

  // ... (email/password validation) ...

  const exists = await User.findOne({ email: normalizedEmail });
  if (exists) {
   return res.status(409).json({ message: "Email already registered" });
  }

  const userRole = role || "user";
```

```
    const newUser = new User({
      username: username.trim(),
      email: normalizedEmail,
      password: trimmedPassword,
      role: userRole
    });

    await newUser.save();

    res.status(201).json({ message: `User registered successfully as ${userRole}` });
  } catch (err) {
    res.status(500).json({ error: "Registration failed", details: err.message });
  }
});

// ===== LOGIN =====
router.post("/login", async (req, res) => {
  try {
    const { email, password } = req.body;
    // ... (validation logic remains the same) ...

    const normalizedEmail = email.trim().toLowerCase();
    const trimmedPassword = password.trim();

    const user = await User.findOne({ email: normalizedEmail });

    // ⚠️ Comparing plaintext password for testing
    if (!user || user.password !== trimmedPassword) {
      return res.status(401).json({ message: "Invalid email or password" });
    }

    res.status(200).json({
      message: "Login successful",
      user: {
        id: user._id,
        username: user.username,
        email: user.email,
        role: user.role
      }
    });
```

```
 } catch (err) {
   res.status(500).json({ error: "Login failed", details: err.message });
 }
});

export default router;
```

blogPostModel.js

```
import mongoose from "mongoose";

const blogPostSchema = new mongoose.Schema({
 title: String,
 content: String,
 author: String,
 createdAt: { type: Date, default: Date.now }
});

export default mongoose.model("BlogPost", blogPostSchema);
```

blogRoutes.js

```
import express from "express";
import BlogPost from "./blogPostModel.js"; // ⬅️ FIX: Added .js extension

const router = express.Router();

// ===== CREATE BLOG POST =====
router.post("/", async (req, res) => {
// ... (Logic remains the same) ...
});

// ===== GET ALL POSTS =====
router.get("/", async (req, res) => {
// ... (Logic remains the same) ...
});

// ===== GET SINGLE POST =====
router.get("/:id", async (req, res) => {
// ... (Logic remains the same) ...
```

```
});

export default router;
```

commentModel.js

```
import mongoose from "mongoose";

const commentSchema = new mongoose.Schema(
 {
   postId: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
    ref: 'BlogPost'
   },
   user: {
    type: String,
    required: true
   },
   text: {
    type: String,
    required: true
   }
 },
 { timestamps: true }
);

export default mongoose.model("Comment", commentSchema);
```

commentRoutes.js

```
import express from "express";
import Comment from "./commentModel.js"; // ⬅️ Needs the .js extension

const router = express.Router();

// ===== GET COMMENTS BY POST ID =====
router.get("/:postId", async (req, res) => {
 try {
   const { postId } = req.params;
```

```
    const comments = await Comment.find({ postId: postId }).sort({ createdAt: 1 });
    res.status(200).json(comments);
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch comments", details: err.message });
  }
});

// ===== CREATE NEW COMMENT =====
router.post("/", async (req, res) => {
  try {
    const { postId, user, text } = req.body;

    if (!postId || !user || !text) {
      return res.status(400).json({ message: "All fields (postId, user, text) are required" });
    }

    const newComment = new Comment({
      postId,
      user: user.trim(),
      text: text.trim(),
    });

    await newComment.save();
    res.status(201).json({ message: "Comment posted successfully", comment:
newComment });
  } catch (err) {
    res.status(500).json({ error: "Failed to post comment", details: err.message });
  }
});

export default router; // 🚨 CRITICAL FIX: Ensures server.js can import it
```

server.js

```
import express from "express";
import mongoose from "mongoose";
import cors from "cors";
import dotenv from "dotenv";
import path from "path";
import { fileURLToPath } from "url";
```

```
// ===== Fix __dirname for ES modules & load .env properly =====
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

dotenv.config({ path: path.join(__dirname, ".env") });

// Test: uncomment to verify dotenv is loaded
// console.log("MONGO_URI =", process.env.MONGO_URI);

// ===== Routes Imports =====
import authRoutes from "./authRoutes.js";
import blogRoutes from "./blogRoutes.js";
import commentRoutes from "./commentRoutes.js";

const app = express();

// ===== Middleware (CORS & Body Parsing) =====
const corsOptions = {
  origin: "http://localhost:3000",
  methods: ["GET", "POST", "PUT", "DELETE"],
  credentials: true,
};

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

// ===== Routes =====
app.get("/", (req, res) => {
  res.send("API is running...");
});

app.use("/api/auth", authRoutes);
app.use("/api/blog", blogRoutes);
app.use("/api/comments", commentRoutes);

// ===== MongoDB Connection & Debugging =====
const MONGO_URI = process.env.MONGO_URI;

if (!MONGO_URI) {
  console.error(
```

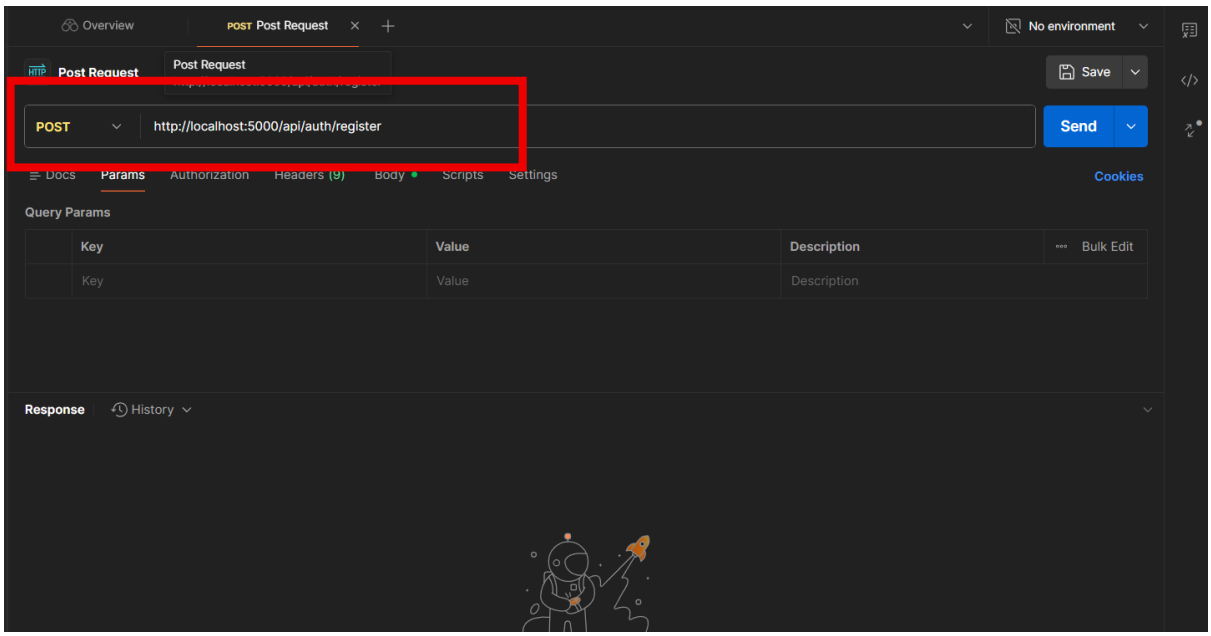
```
    "❌ Fatal: MONGO_URI is not defined in environment variables or .env file."
  );
  process.exit(1);
}

const db = mongoose.connection;
db.on("error", (err) => {
  console.error("❌ Mongoose default connection error: " + err);
});
db.on("disconnected", () => {
  console.log("⚠️ Mongoose default connection disconnected");
});

mongoose
  .connect(MONGO_URI)
  .then(() => console.log("✅ MongoDB connected successfully to Atlas!"))
  .catch((err) => {
    console.error("❌ MongoDB connection failed during startup:", err.message);
    process.exit(1);
  });

// ===== Server Startup =====
const PORT = process.env.PORT || 5000;

app
  .listen(PORT, () => {
    console.log(`🚀 Server running on http://localhost:${PORT}`);
  })
  .on("error", (e) => {
    if (e.code === "EADDRINUSE") {
      console.error(
        `🚨 Port ${PORT} is already in use. Try a different port or stop the conflicting process.`
      );
    } else {
      console.error("❌ Server startup failed:", e.message);
    }
    process.exit(1);
  });
```

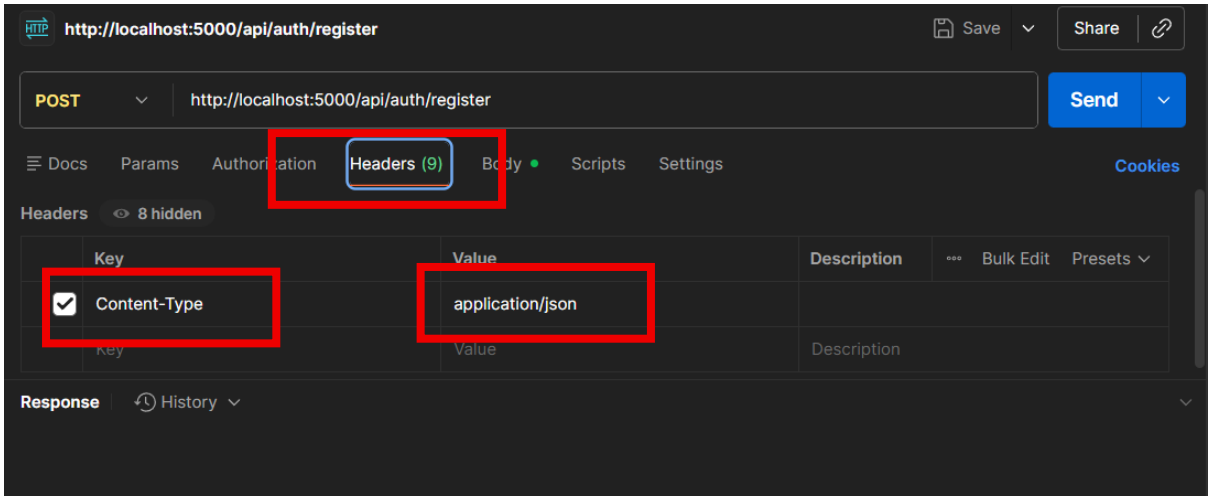
userModel.js

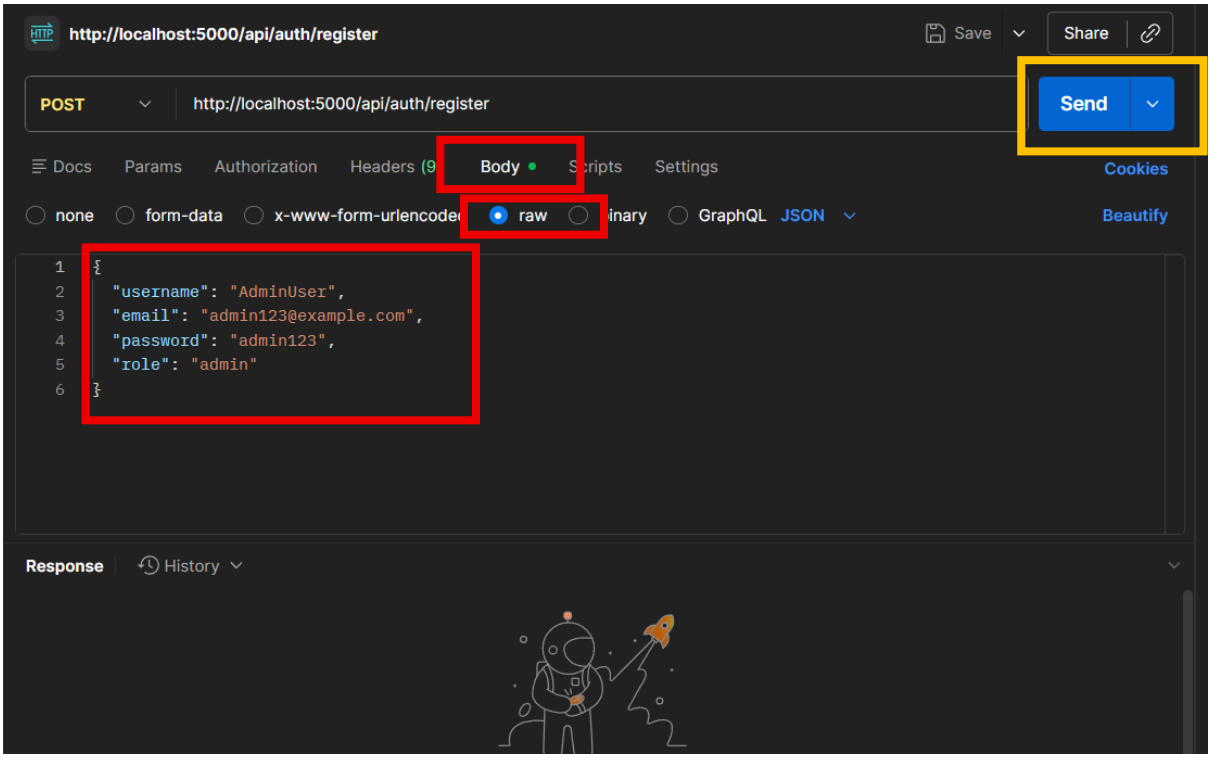
```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
 {
   username: { type: String, required: true, trim: true },
   email: { type: String, required: true, unique: true, trim: true },
   password: { type: String, required: true },
   role: { type: String, default: "user" }
 },
 { timestamps: true }
);

export default mongoose.model("User", userSchema);
```

4. Creating Admin using Postman

1. Open Postman
2. Create a new POST request







3. Check MongoDB for the admin account

```
_id: ObjectId('6930080cb9e90f5a48cf788d')
username : "AdminUser"
email : "admin123@example.com"
password : "admin123"
role : "admin"
createdAt : 2025-12-03T09:51:08.058+00:00
updatedAt : 2025-12-03T09:51:08.058+00:00
__v : 0
```