NAME	Ramendra
UID	23BCS12146
CLASS	622-B

EXPERIMENT 10.1,10.2,10.3

Practice 1

Title
Todo Application with Basic CRUD Operations

Objective
Learn how to design and build a simple full stack application that performs Create, Read, Update, and Delete (CRUD) operations. Understand the basic flow of storing, retrieving, and managing data in a database.

CODE

backend/package.json

```
{
  "name": "todo-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
```

```
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "mongoose": "^7.0.4"
  },
  "devDependencies": {
    "nodemon": "^3.0.0"
  }
}
```

backend/server.js

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
require("dotenv").config();

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

mongoose
  .connect(process.env.MONGO_URI || "mongodb://localhost:27017/todoapp")
  .then(() => console.log("MongoDB connected"))
  .catch(err => console.log(err));

const todoRoutes = require("./routes/todoRoutes");
app.use("/api/todos", todoRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

backend/models/Todo.js

```
const mongoose = require("mongoose");

const todoSchema = new mongoose.Schema({
  text: { type: String, required: true },
  completed: { type: Boolean, default: false },
});
```

```
module.exports = mongoose.model("Todo", todoSchema);
backend/routes/todoRoutes.js
const express = require("express");
const router = express.Router();
const Todo = require("../models/Todo");

// CREATE
router.post("/", async (req, res) => {
  try {
    const todo = new Todo({ text: req.body.text });
    const saved = await todo.save();
    res.json(saved);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// READ
router.get("/", async (req, res) => {
  try {
    const todos = await Todo.find();
    res.json(todos);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// UPDATE
router.put("/:id", async (req, res) => {
  try {
    const updated = await Todo.findByIdAndUpdate(
      req.params.id,
      { completed: req.body.completed },
      { new: true }
    );
    res.json(updated);
  } catch (err) {
```

```
    res.status(500).json({ error: err.message });
  }
});

// DELETE
router.delete("/:id", async (req, res) => {
  try {
    await Todo.findByIdAndDelete(req.params.id);
    res.json({ message: "Todo deleted" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

🎨 2. Frontend Code (React)

frontend/package.json

```
{
  "name": "todo-frontend",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "axios": "^1.3.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

frontend/src/App.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import TodoItem from "./components/TodoItem";
```

```
const API_URL = "http://localhost:5000/api/todos";

function App() {
  const [todos, setTodos] = useState([]);
  const [text, setText] = useState("");

  useEffect(() => {
    fetchTodos();
  }, []);

  const fetchTodos = async () => {
    const res = await axios.get(API_URL);
    setTodos(res.data);
  };

  const addTodo = async () => {
    if (!text.trim()) return;
    await axios.post(API_URL, { text });
    setText("");
    fetchTodos();
  };

  const toggleComplete = async (id, completed) => {
    await axios.put(`${API_URL}/${id}`, { completed: !completed });
    fetchTodos();
  };

  const deleteTodo = async (id) => {
    await axios.delete(`${API_URL}/${id}`);
    fetchTodos();
  };

  return (
    <div style={{ margin: "50px auto", width: "400px", textAlign: "center" }}>
      <h1>Todo List</h1>
      <div>
```

```
        <input
          style={{ padding: "8px", width: "70%" }}
          value={text}
          onChange={(e) => setText(e.target.value)}
          placeholder="Add new todo"
        />
        <button
          style={{ padding: "8px", marginLeft: "5px" }}
          onClick={addTodo}
        >
          Add
        </button>
      </div>
      <ul style={{ listStyle: "none", padding: 0, marginTop: "20px" }}>
        {todos.map((todo) => (
          <TodoItem
            key={todo._id}
            todo={todo}
            onToggle={toggleComplete}
            onDelete={deleteTodo}
          />
        ))}
      </ul>
    </div>
  );
}

export default App;
```

frontend/src/components/TodoItem.js

```
import React from "react";

export default function TodoItem({ todo, onToggle, onDelete }) {
  return (
    <li
      style={{
        display: "flex",
        justifyContent: "space-between",
```

```
        alignItems: "center",
        background: "#f8f8f8",
        padding: "10px",
        marginBottom: "8px",
        borderRadius: "6px",
      }}
    >
      <span
        style={{
          textDecoration: todo.completed ? "line-through" : "none",
          flexGrow: 1,
          textAlign: "left",
          paddingLeft: "8px",
        }}
        onClick={() => onToggle(todo._id, todo.completed)}
      >
        {todo.text}
      </span>
      <button
        style={{
          background: "red",
          color: "white",
          border: "none",
          borderRadius: "5px",
          padding: "5px 10px",
        }}
        onClick={() => onDelete(todo._id)}
      >
        Delete
      </button>
    </li>
  );
}
```

frontend/public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <title>Todo App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

frontend/src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

Practice 2

Title

Blog Platform with Comments and User Profiles

Objective

Develop a more advanced application that manages dynamic user-generated content. Understand user authentication, relationships between users and posts, and data modeling for comments**.**

CODE

backend/package.json

```
{
  "name": "blog-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
```

```
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.0",
    "mongoose": "^7.0.4"
  },
  "devDependencies": {
    "nodemon": "^3.0.0"
  }
}
```

---

backend/server.js

```
require("dotenv").config();
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const app = express();
app.use(cors());
app.use(express.json());

mongoose.connect(process.env.MONGO_URI || "mongodb://localhost:27017/blogdb")
  .then(() => console.log("MongoDB Connected"))
  .catch(err => console.log(err));

app.use("/api/auth", require("./routes/auth"));
app.use("/api/posts", require("./routes/posts"));
app.use("/api/comments", require("./routes/comments"));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

---

backend/models/User.js

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
```

```
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  passwordHash: { type: String, required: true },
  bio: { type: String, default: "" },
  avatar: { type: String, default: "" }
}, { timestamps: true });

module.exports = mongoose.model("User", userSchema);
```

---

backend/models/Post.js

```
const mongoose = require("mongoose");

const postSchema = new mongoose.Schema({
  author: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  title: { type: String, required: true },
  body: { type: String, required: true }
}, { timestamps: true });

module.exports = mongoose.model("Post", postSchema);
```

---

backend/models/Comment.js

```
const mongoose = require("mongoose");

const commentSchema = new mongoose.Schema({
  post: { type: mongoose.Schema.Types.ObjectId, ref: "Post", required: true },
  author: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  text: { type: String, required: true }
}, { timestamps: true });

module.exports = mongoose.model("Comment", commentSchema);
```

---

backend/middleware/auth.js

```
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {
  const token = req.header("Authorization")?.split(" ")[1];
  if (!token) return res.status(401).json({ message: "Access denied" });
```

```
  try {
    const verified = jwt.verify(token, process.env.JWT_SECRET);
    req.user = verified;
    next();
  } catch (err) {
    res.status(400).json({ message: "Invalid token" });
  }
};
```

---

backend/routes/auth.js

```
const express = require("express");
const router = express.Router();
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const User = require("../models/User");

// Register
router.post("/register", async (req, res) => {
  const { username, email, password } = req.body;
  const existing = await User.findOne({ email });
  if (existing) return res.status(400).json({ message: "User already exists" });

  const hash = await bcrypt.hash(password, 10);
  const user = new User({ username, email, passwordHash: hash });
  await user.save();

  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET);
  res.json({ token, user: { id: user._id, username, email } });
});

// Login
router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(400).json({ message: "Invalid credentials" });
```

```
  const valid = await bcrypt.compare(password, user.passwordHash);
  if (!valid) return res.status(400).json({ message: "Invalid credentials" });

  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET);
  res.json({ token, user: { id: user._id, username: user.username, email } });
});

module.exports = router;
```

---

backend/routes/posts.js

```
const express = require("express");
const router = express.Router();
const Post = require("../models/Post");
const auth = require("../middleware/auth");

// Create post
router.post("/", auth, async (req, res) => {
  const post = new Post({ author: req.user.id, title: req.body.title, body: req.body.body });
  await post.save();
  res.json(post);
});

// Get all posts
router.get("/", async (req, res) => {
  const posts = await Post.find().populate("author", "username email");
  res.json(posts);
});

// Get single post
router.get("/:id", async (req, res) => {
  const post = await Post.findById(req.params.id).populate("author", "username email");
  res.json(post);
});

// Update post
router.put("/:id", auth, async (req, res) => {
  const post = await Post.findById(req.params.id);
```

```
  if (!post) return res.status(404).json({ message: "Not found" });
  if (post.author.toString() !== req.user.id) return res.status(403).json({ message: "Forbidden" });
  post.title = req.body.title;
  post.body = req.body.body;
  await post.save();
  res.json(post);
});

// Delete post
router.delete("/:id", auth, async (req, res) => {
  const post = await Post.findById(req.params.id);
  if (!post) return res.status(404).json({ message: "Not found" });
  if (post.author.toString() !== req.user.id) return res.status(403).json({ message: "Forbidden" });
  await post.remove();
  res.json({ message: "Deleted" });
});

module.exports = router;
```

---

backend/routes/comments.js

```
const express = require("express");
const router = express.Router();
const Comment = require("../models/Comment");
const auth = require("../middleware/auth");

// Add comment
router.post("/:postId", auth, async (req, res) => {
  const comment = new Comment({ post: req.params.postId, author: req.user.id, text: req.body.text });
  await comment.save();
  const populated = await comment.populate("author", "username");
  res.json(populated);
});

// Get comments
```

```
router.get("/:postId", async (req, res) => {
  const comments = await Comment.find({ post: req.params.postId }).populate("author",
"username");
  res.json(comments);
});

module.exports = router;
```

🎨 Frontend Implementation (React)

frontend/package.json

```
{
  "name": "blog-frontend",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "axios": "^1.4.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.1",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

---

frontend/src/api.js

```
import axios from "axios";
const API = axios.create({ baseURL: "http://localhost:5000/api" });
API.interceptors.request.use(config => {
  const token = localStorage.getItem("token");
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});
export default API;
```

---

frontend/src/components/Navbar.js

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

export default function Navbar() {
  const nav = useNavigate();
  const user = JSON.parse(localStorage.getItem("user"));
  const logout = () => { localStorage.clear(); nav("/"); };
  return (
    <nav style={{ background: "#222", padding: "10px", color: "white" }}>
      <Link to="/" style={{ color: "white", marginRight: "10px" }}>Home</Link>
      {user ? (
        <>
          <Link to="/create" style={{ color: "white", marginRight: "10px" }}>Create Post</Link>
          <span>{user.username}</span>
          <button onClick={logout} style={{ marginLeft: "10px" }}>Logout</button>
        </>
      ) : (
        <>
          <Link to="/login" style={{ color: "white", marginRight: "10px" }}>Login</Link>
          <Link to="/register" style={{ color: "white" }}>Register</Link>
        </>
      )}
    </nav>
  );
}
```

---

frontend/src/pages/Home.js

```
import React, { useEffect, useState } from "react";
import API from "../api";
import { Link } from "react-router-dom";

export default function Home() {
  const [posts, setPosts] = useState([]);
  useEffect(() => { API.get("/posts").then(res => setPosts(res.data)); }, []);
  return (
```

```
    <div style={{ padding: "20px" }}>
      <h2>All Posts</h2>
      {posts.map(post => (
        <div key={post._id} style={{ marginBottom: "20px", borderBottom: "1px solid #ccc" }}>
          <h3><Link to={`/post/${post._id}`}>{post.title}</Link></h3>
          <p>by {post.author.username}</p>
        </div>
      ))}
    </div>
  );
}
```

---

frontend/src/pages/Login.js

```
import React, { useState } from "react";
import API from "../api";
import { useNavigate } from "react-router-dom";

export default function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const nav = useNavigate();

  const handleLogin = async () => {
    const res = await API.post("/auth/login", { email, password });
    localStorage.setItem("token", res.data.token);
    localStorage.setItem("user", JSON.stringify(res.data.user));
    nav("/");
  };

  return (
    <div style={{ padding: "20px" }}>
      <h2>Login</h2>
      <input placeholder="Email" onChange={e => setEmail(e.target.value)} /><br/>
      <input placeholder="Password" type="password" onChange={e => setPassword(e.target.value)} /><br/>
      <button onClick={handleLogin}>Login</button>
```

```
    </div>
  );
}
```

---

frontend/src/pages/Register.js

```
import React, { useState } from "react";
import API from "../api";
import { useNavigate } from "react-router-dom";

export default function Register() {
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const nav = useNavigate();

  const handleRegister = async () => {
    const res = await API.post("/auth/register", { username, email, password });
    localStorage.setItem("token", res.data.token);
    localStorage.setItem("user", JSON.stringify(res.data.user));
    nav("/");
  };

  return (
    <div style={{ padding: "20px" }}>
      <h2>Register</h2>
      <input placeholder="Username" onChange={e => setUsername(e.target.value)}
/><br/>
      <input placeholder="Email" onChange={e => setEmail(e.target.value)} /><br/>
      <input placeholder="Password" type="password" onChange={e =>
setPassword(e.target.value)} /><br/>
      <button onClick={handleRegister}>Register</button>
    </div>
  );
}
```

---

frontend/src/pages/CreatePost.js

```
import React, { useState } from "react";
```

```
import API from "../api";
import { useNavigate } from "react-router-dom";

export default function CreatePost() {
  const [title, setTitle] = useState("");
  const [body, setBody] = useState("");
  const nav = useNavigate();

  const handleCreate = async () => {
    await API.post("/posts", { title, body });
    nav("/");
  };

  return (
    <div style={{ padding: "20px" }}>
      <h2>New Post</h2>
      <input placeholder="Title" onChange={e => setTitle(e.target.value)} /><br/>
      <textarea placeholder="Body" rows="10" onChange={e => setBody(e.target.value)}
/><br/>
      <button onClick={handleCreate}>Post</button>
    </div>
  );
}
```

---

frontend/src/pages/PostView.js

```
import React, { useEffect, useState } from "react";
import API from "../api";
import { useParams } from "react-router-dom";

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

  useEffect(() => {
    API.get(`/posts/${id}`).then(res => setPost(res.data));
```

```
  API.get(`/comments/${id}`).then(res => setComments(res.data));
 }, [id]);

 const addComment = async () => {
  const res = await API.post(`/comments/${id}`, { text });
  setComments([...comments, res.data]);
  setText("");
 };

 return (
  <div style={{ padding: "20px" }}>
   {post && (
    <>
     <h2>{post.title}</h2>
     <p>by {post.author.username}</p>
     <p>{post.body}</p>
     <hr />
     <h3>Comments</h3>
     {comments.map(c => (
      <div key={c._id}>
       <strong>{c.author.username}</strong>: {c.text}
      </div>
     ))}
     <textarea value={text} onChange={e => setText(e.target.value)} rows="3" />
     <br />
     <button onClick={addComment}>Add Comment</button>
    </>
   )}
  </div>
 );
}
```

Practice 3

Title
Social Media App with Posts, Authentication, and AWS Deployment

Objective
Build a complex real-world project that combines frontend, backend, user authentication, and cloud deployment. Learn how to make scalable apps, deploy them, and handle real user interactions.

CODE

backend/package.json

```
{
  "name": "social-media-backend",
  "version": "1.0.0",
  "main": "src/server.js",
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon src/server.js"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.4.0",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "multer": "^1.4.5-lts.1",
    "pg": "^8.11.0",
    "sequelize": "^6.35.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.0"
  }
}
```

---

backend/src/server.js

```
require("dotenv").config();
const express = require("express");
const cors = require("cors");
const { sequelize } = require("./models/User");
const authRoutes = require("./routes/auth");
const postRoutes = require("./routes/posts");
const commentRoutes = require("./routes/comments");

const app = express();
app.use(cors());
app.use(express.json());
app.use("/uploads", express.static("uploads"));

app.use("/api/auth", authRoutes);
app.use("/api/posts", postRoutes);
app.use("/api/comments", commentRoutes);

sequelize.sync().then(() => console.log("Database synced"));
app.listen(process.env.PORT || 5000, () => console.log("Server running..."));
```

---

backend/src/models/User.js

```
const { Sequelize, DataTypes } = require("sequelize");
const sequelize = new Sequelize(process.env.DATABASE_URL);

const User = sequelize.define("User", {
  username: { type: DataTypes.STRING, unique: true },
  email: { type: DataTypes.STRING, unique: true },
  passwordHash: { type: DataTypes.STRING },
});

module.exports = { User, sequelize };
```

---

backend/src/models/Post.js

```
const { Sequelize, DataTypes } = require("sequelize");
const { sequelize } = require("./User");
```

```
const Post = sequelize.define("Post", {
  title: DataTypes.STRING,
  content: DataTypes.TEXT,
  imageUrl: DataTypes.STRING,
  likes: { type: DataTypes.INTEGER, defaultValue: 0 },
});

module.exports = { Post };
```

---

backend/src/models/Comment.js

```
const { Sequelize, DataTypes } = require("sequelize");
const { sequelize } = require("./User");

const Comment = sequelize.define("Comment", {
  text: DataTypes.TEXT,
});

module.exports = { Comment };
```

---

backend/src/routes/auth.js

```
const express = require("express");
const router = express.Router();
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const { User } = require("../models/User");

router.post("/register", async (req, res) => {
  const { username, email, password } = req.body;
  const hash = await bcrypt.hash(password, 10);
  const user = await User.create({ username, email, passwordHash: hash });
  res.json(user);
});

router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ where: { email } });
  if (!user || !(await bcrypt.compare(password, user.passwordHash)))
```

```
    return res.status(401).json({ message: "Invalid credentials" });

  const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expiresIn: "7d" });
  res.json({ token, user });
});

module.exports = router;
```

---

backend/src/routes/posts.js

```
const express = require("express");
const router = express.Router();
const { Post } = require("../models/Post");
const auth = require("../middleware/auth");
const multer = require("multer");

const upload = multer({ dest: "uploads/" });

router.post("/", auth, upload.single("image"), async (req, res) => {
  const post = await Post.create({
    title: req.body.title,
    content: req.body.content,
    imageUrl: req.file ? `/uploads/${req.file.filename}` : null,
    UserId: req.user.id,
  });
  res.json(post);
});

router.get("/", async (req, res) => {
  const posts = await Post.findAll({ order: [["createdAt", "DESC"]] });
  res.json(posts);
});

router.put("/:id", auth, async (req, res) => {
  const post = await Post.findByPk(req.params.id);
  if (!post || post.UserId !== req.user.id) return res.status(403).json({ message:
"Forbidden" });
  await post.update(req.body);
```

```
  res.json(post);
});

router.delete("/:id", auth, async (req, res) => {
  const post = await Post.findByPk(req.params.id);
  if (!post || post.UserId !== req.user.id) return res.status(403).json({ message: "Forbidden" });
  await post.destroy();
  res.json({ message: "Deleted" });
});

module.exports = router;
```

---

backend/src/middleware/auth.js

```
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {
  const token = req.header("Authorization")?.split(" ")[1];
  if (!token) return res.status(401).json({ message: "No token" });
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch {
    res.status(401).json({ message: "Invalid token" });
  }
};
```

🎨 Frontend (React)

frontend/package.json

```
{
  "name": "social-media-frontend",
  "version": "1.0.0",
  "dependencies": {
    "axios": "^1.3.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
```

```
    "react-router-dom": "^6.14.1",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

---

frontend/src/api.js

```
import axios from "axios";
const API = axios.create({ baseURL: "http://<YOUR-EC2-PUBLIC-IP>:5000/api" });
API.interceptors.request.use(config => {
  const token = localStorage.getItem("token");
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});
export default API;
```

---

frontend/src/pages/Home.js

```
import React, { useEffect, useState } from "react";
import API from "../api";

export default function Home() {
  const [posts, setPosts] = useState([]);
  useEffect(() => { API.get("/posts").then(r => setPosts(r.data)); }, []);
  return (
    <div style={{ padding: "20px" }}>
      <h2>News Feed</h2>
      {posts.map(p => (
        <div key={p.id} style={{ marginBottom: "20px" }}>
          <h3>{p.title}</h3>
          {p.imageUrl && <img src={p.imageUrl} alt="post" width="200" />}
          <p>{p.content}</p>
        </div>
      ))}
    </div>
```

```
  );
}
```