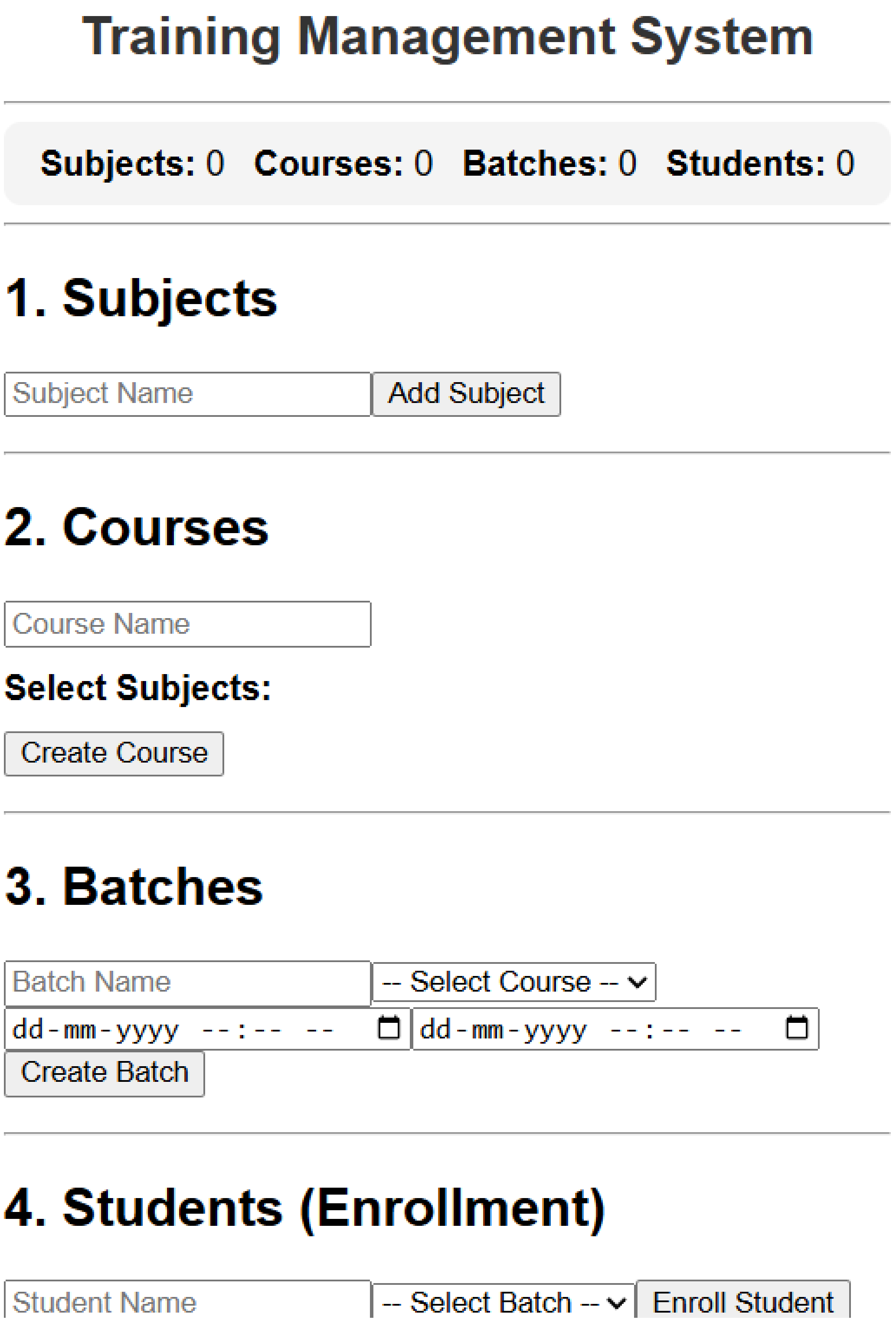
```
import React, { useReducer, useEffect, useState } from "react";

/* -------------------- Utilities -------------------- */

// Replaced uuid library with native browser crypto for zero-dependency
running
const generateId = () => crypto.randomUUID();

const normalize = (s) => (s ? s.trim().toLowerCase() : "");

const isTimeValid = (start, end) => new Date(start) < new Date(end);

const overlap = (a1, a2, b1, b2) =>
  new Date(a1) < new Date(b2) && new Date(b1) < new Date(a2);

/* -------------------- Store -------------------- */

const initialState = {
  subjects: {},
  courses: {},
  batches: {},
  students: {}
};

function reducer(state, action) {
  switch (action.type) {
    case "LOAD":
```

```
  return action.payload || initialState;

case "ADD_SUBJECT": {
  const id = generateId();
  return {
    ...state,
    subjects: { ...state.subjects, [id]: { id, name: action.name } }
  };
}

case "DELETE_SUBJECT": {
  const used = Object.values(state.courses).some((c) =>
    c.subjectIds.includes(action.id)
  );
  if (used) {
    alert("Cannot delete: Subject is part of an existing course.");
    return state;
  }
  const copy = { ...state.subjects };
  delete copy[action.id];
  return { ...state, subjects: copy };
}

case "ADD_COURSE": {
  const id = generateId();
  return {
        ...state,
        courses: {
          ...state.courses,
          [id]: { id, name: action.name, subjectIds: action.subjectIds }
        }
      };
    }

    case "DELETE_COURSE": {
      const used = Object.values(state.batches).some((b) => b.courseId ===
action.id);
      if (used) {
        alert("Cannot delete: Course is assigned to a batch.");
        return state;
      }
      const copy = { ...state.courses };
      delete copy[action.id];
      return { ...state, courses: copy };
    }

    case "ADD_BATCH": {
      const id = generateId();
      return {
        ...state,
        batches: { ...state.batches, [id]: { id, ...action.payload } }
      };
    }
    case "DELETE_BATCH": {
      const used = Object.values(state.students).some((s) => s.batchId ===
action.id);
      if (used) {
        alert("Cannot delete: Batch has enrolled students.");
        return state;
      }
      const copy = { ...state.batches };
      delete copy[action.id];
      return { ...state, batches: copy };
    }

    case "ADD_STUDENT": {
      const id = generateId();
      return {
        ...state,
        students: { ...state.students, [id]: { id, ...action.payload } }
      };
    }

    case "DELETE_STUDENT": {
      const copy = { ...state.students };
      delete copy[action.id];
      return { ...state, students: copy };
    }
    default:
      return state;
  }
}

/* -------------------- Main App -------------------- */

export default function App() {
  const [state, dispatch] = useReducer(reducer, initialState);

  // Load from LocalStorage
  useEffect(() => {
    const saved = localStorage.getItem("training_app_data");
    if (saved) {
      try {
        dispatch({ type: "LOAD", payload: JSON.parse(saved) });
      } catch (e) {
        console.error("Failed to load data", e);
      }
    }
  }, []);

  // Save to LocalStorage
  useEffect(() => {
    if (state !== initialState) {
      localStorage.setItem("training_app_data", JSON.stringify(state));
    }
  }, [state]);

  return (
    <div style={{ padding: "20px", fontFamily: "Arial, sans-serif", maxWidth:
"800px", margin: "0 auto" }}>
      <h1 style={{ textAlign: "center", color: "#333" }}>Training Management
System</h1>
      <hr />
      <Dashboard state={state} />
      <hr />
      <Subjects state={state} dispatch={dispatch} />
      <hr />
      <Courses state={state} dispatch={dispatch} />
      <hr />
      <Batches state={state} dispatch={dispatch} />
      <hr />
      <Students state={state} dispatch={dispatch} />
    </div>
  );
}

/* -------------------- Sub-Components -------------------- */

function Dashboard({ state }) {
  return (
    <div style={{ display: "flex", justifyContent: "space-around",
backgroundColor: "#f4f4f4", padding: "10px", borderRadius: "8px" }}>
      <div><strong>Subjects:</strong> {Object.keys(state.subjects).length}</div>
      <div><strong>Courses:</strong> {Object.keys(state.courses).length}</div>
      <div><strong>Batches:</strong> {Object.keys(state.batches).length}</div>
      <div><strong>Students:</strong>
{Object.keys(state.students).length}</div>
    </div>
  );
}

function Subjects({ state, dispatch }) {
  const [name, setName] = useState("");

  const add = () => {
    if (!name.trim()) return alert("Name required");
    const exists = Object.values(state.subjects).some(
      (s) => normalize(s.name) === normalize(name)
    );
    if (exists) return alert("Duplicate subject");
    dispatch({ type: "ADD_SUBJECT", name: name.trim() });
    setName("");
  };

  return (
    <div>
      <h2>1. Subjects</h2>
      <input value={name} onChange={(e) => setName(e.target.value)}
placeholder="Subject Name" />
      <button onClick={add}>Add Subject</button>
      <ul>
        {Object.values(state.subjects).map((s) => (
          <li key={s.id}>
            {s.name} <button onClick={() => dispatch({ type: "DELETE_SUBJECT", id:
s.id })}>x</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

function Courses({ state, dispatch }) {
  const [name, setName] = useState("");
  const [selected, setSelected] = useState([]);

  const toggle = (id) =>
    setSelected((p) => (p.includes(id) ? p.filter((x) => x !== id) : [...p, id]));

  const add = () => {
    if (!name.trim()) return alert("Name required");
    if (selected.length < 2) return alert("Select at least 2 subjects");
    dispatch({ type: "ADD_COURSE", name: name.trim(), subjectIds: selected });
    setName("");
    setSelected([]);
  };

  return (
    <div>
      <h2>2. Courses</h2>
      <input value={name} onChange={(e) => setName(e.target.value)}
placeholder="Course Name" />
      <div style={{ margin: "10px 0" }}>
        <strong>Select Subjects:</strong><br/>
        {Object.values(state.subjects).map((s) => (
          <label key={s.id} style={{ marginRight: "10px" }}>
            <input type="checkbox" checked={selected.includes(s.id)} onChange={()
=> toggle(s.id)} />
            {s.name}
          </label>
        ))}
      </div>
      <button onClick={add}>Create Course</button>
      <ul>
        {Object.values(state.courses).map((c) => (
          <li key={c.id}>
            {c.name} ({c.subjectIds.length} subjects)
            <button onClick={() => dispatch({ type: "DELETE_COURSE", id: c.id
})}>x</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

function Batches({ state, dispatch }) {
  const [name, setName] = useState("");
  const [courseId, setCourseId] = useState("");
  const [start, setStart] = useState("");
  const [end, setEnd] = useState("");

  const add = () => {
    if (!name || !courseId || !start || !end) return alert("All fields required");
    if (!isTimeValid(start, end)) return alert("End time must be after start time");
    dispatch({ type: "ADD_BATCH", payload: { name, courseId, start, end } });
    setName(""); setCourseId(""); setStart(""); setEnd("");
  };

  return (
    <div>
      <h2>3. Batches</h2>
      <input placeholder="Batch Name" value={name} onChange={(e) =>
setName(e.target.value)} />
      <select value={courseId} onChange={(e) => setCourseId(e.target.value)}>
        <option value="">-- Select Course --</option>
        {Object.values(state.courses).map((c) => (
          <option key={c.id} value={c.id}>{c.name}</option>
        ))}
      </select>
      <input type="datetime-local" value={start} onChange={(e) =>
setStart(e.target.value)} />
      <input type="datetime-local" value={end} onChange={(e) =>
setEnd(e.target.value)} />
      <button onClick={add}>Create Batch</button>
      <ul>
        {Object.values(state.batches).map((b) => (
          <li key={b.id}>
            {b.name} ({state.courses[b.courseId]?.name})
            <button onClick={() => dispatch({ type: "DELETE_BATCH", id: b.id
})}>x</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

function Students({ state, dispatch }) {
  const [name, setName] = useState("");
  const [batchId, setBatchId] = useState("");

  const add = () => {
    if (!name || !batchId) return alert("Name and Batch required");
    const batch = state.batches[batchId];
    // Conflict Check: Is student already in a batch that overlaps with this one?
    const conflict = Object.values(state.students).some((s) => {
      if (s.name.toLowerCase() !== name.toLowerCase()) return false;
      const existingBatch = state.batches[s.batchId];
      return overlap(batch.start, batch.end, existingBatch.start,
existingBatch.end);
    });

    if (conflict) return alert("This student has a scheduling conflict with another
batch!");

    dispatch({ type: "ADD_STUDENT", payload: { name, batchId } });
    setName(""); setBatchId("");
  };

  return (
    <div>
      <h2>4. Students (Enrollment)</h2>
      <input value={name} onChange={(e) => setName(e.target.value)}
placeholder="Student Name" />
      <select value={batchId} onChange={(e) => setBatchId(e.target.value)}>
        <option value="">-- Select Batch --</option>
        {Object.values(state.batches).map((b) => (
          <option key={b.id} value={b.id}>{b.name}
({state.courses[b.courseId]?.name})</option>
        ))}
      </select>
      <button onClick={add}>Enroll Student</button>
      <ul>
        {Object.values(state.students).map((s) => (
          <li key={s.id}>
            {s.name} - Enrolled in {state.batches[s.batchId]?.name}
            <button onClick={() => dispatch({ type: "DELETE_STUDENT", id: s.id
})}>x</button>
          </li>
        ))}
      </ul>
    </div>
  );
```

OUT PUT:



Training Management System

Subjects: 0 Courses: 0 Batches: 0 Students: 0

1. Subjects

Subject Name | Add Subject

2. Courses

Course Name

Select Subjects:

Create Course

3. Batches

Batch Name | -- Select Course --

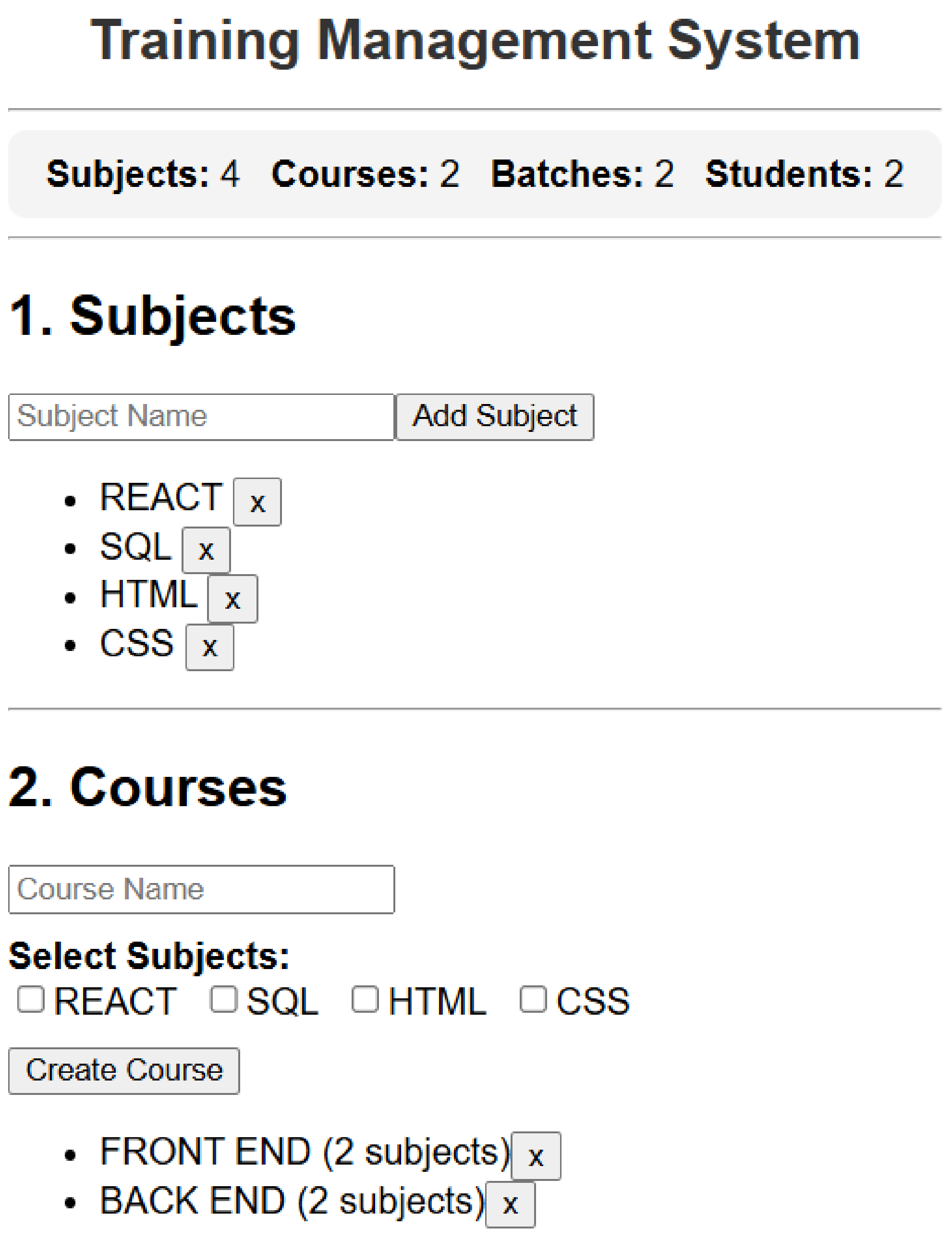
dd-mm-yyyy --:-- -- | dd-mm-yyyy --:-- --

Create Batch

4. Students (Enrollment)

Student Name | -- Select Batch -- | Enroll Student

FINAL OUTPUT:



Training Management System

Subjects: 4 **Courses:** 2 **Batches:** 2 **Students:** 2

1. Subjects

Subject Name | Add Subject

- REACT x
- SQL x
- HTML x
- CSS x

2. Courses

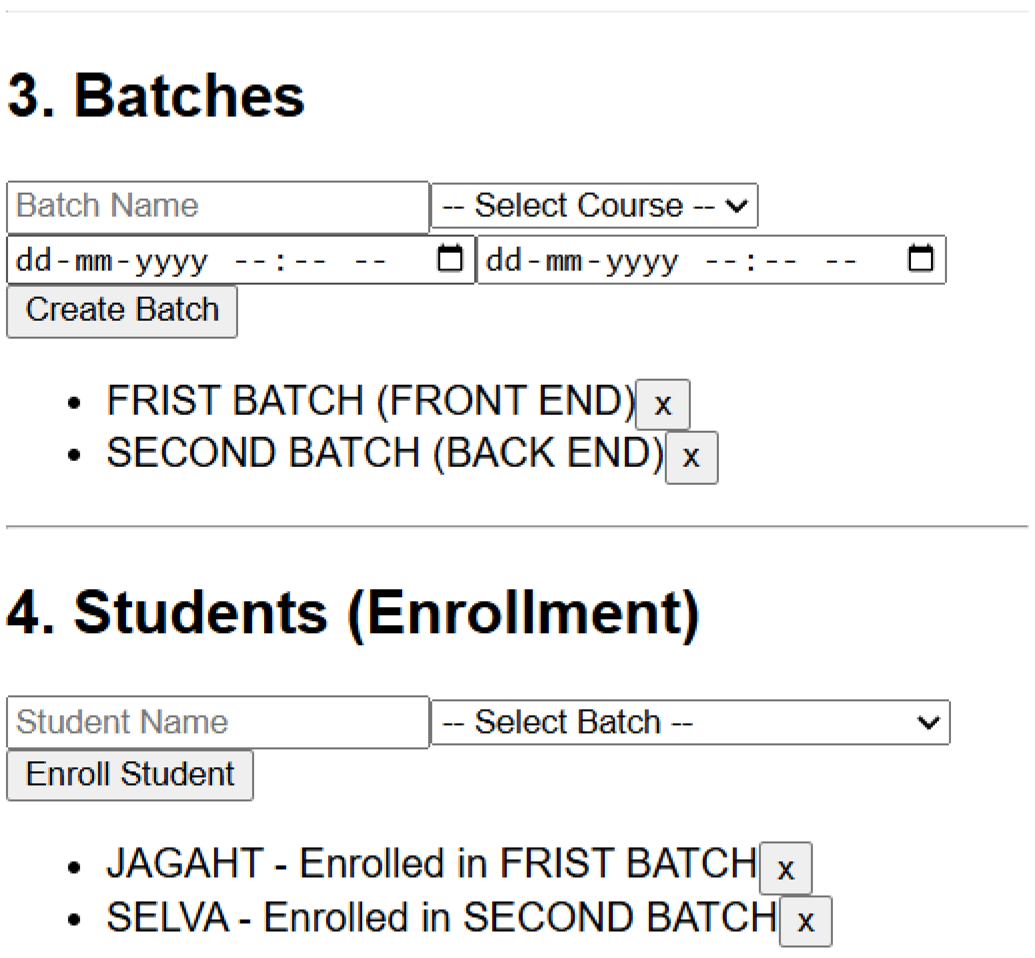
Course Name

Select Subjects:

☐ REACT ☐ SQL ☐ HTML ☐ CSS

Create Course

- FRONT END (2 subjects) x
- BACK END (2 subjects) x



3. Batches

Batch Name | -- Select Course --
dd-mm-yyyy --:-- -- | dd-mm-yyyy --:-- --
Create Batch

- FRIST BATCH (FRONT END) x
- SECOND BATCH (BACK END) x

4. Students (Enrollment)

Student Name | -- Select Batch --
Enroll Student

- JAGAHT - Enrolled in FRIST BATCH x
- SELVA - Enrolled in SECOND BATCH x

THANK YOU