LMS Micro-service App – Consolidated Codebase

Below is the *complete* repository layout followed by every source file exactly where it belongs. You can copy-paste this into a fresh folder and run `npm install` inside each **server** and the **lms-client** to get started.

Directory Tree

```
lms-app/
├── shared.env
├── loginserver/
│   ├── .env
│   ├── server.js
│   └── models/
│       └── Credential.js
├── studentserver/
│   ├── .env
│   ├── server.js
│   └── models/
│       ├── Student.js
│       └── Course.js
├── adminserver/
│   ├── .env
│   ├── server.js
│   └── models/
│       └── Admin.js
├── chatserver/
│   ├── server.js
│   └── chat_logs/            # created at runtime
└── lms-client/
    ├── package.json
    └── src/
        ├── context/
        │   └── UserContext.js
        └── pages/
            ├── Login.js
            ├── StudentHome.js
            ├── StudentModule.js
            ├── Profile.js
            ├── Chat.js
            ├── AdminHome.js
            ├── AdminProfile.js
```

```
        ├── AdminBatch.js
        └── AdminChat.js
```

Environment files

Share the **same** secrets by copying `shared.env` into every server folder as `.env` (already shown below for convenience).

`shared.env` ***(root)***

```
ACCESS_TOKEN_SECRET=access_secret_123
REFRESH_TOKEN_SECRET=refresh_secret_123
MONGO_URI=mongodb://localhost:27017/lms
```

`loginserver/.env` ***(identical copy)***

```
ACCESS_TOKEN_SECRET=access_secret_123
REFRESH_TOKEN_SECRET=refresh_secret_123
MONGO_URI=mongodb://localhost:27017/lms
```

`studentserver/.env` ***(identical copy)***

```
ACCESS_TOKEN_SECRET=access_secret_123
REFRESH_TOKEN_SECRET=refresh_secret_123
MONGO_URI=mongodb://localhost:27017/lms
```

`adminserver/.env` ***(identical copy)***

```
ACCESS_TOKEN_SECRET=access_secret_123
REFRESH_TOKEN_SECRET=refresh_secret_123
MONGO_URI=mongodb://localhost:27017/lms
```

Auth Service – loginserver

`loginserver/models/Credential.js`

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

```
const CredentialSchema = new mongoose.Schema({
  name: String,
  username: { type: String, unique: true },
  password: String,
  role: { type: String, enum: ['student', 'admin'] },
  refresh_token: String
});

module.exports = mongoose.model('Credential', CredentialSchema);
```

loginserver/server.js

```
const express = require('express');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const Credential = require('./models/Credential');
require('dotenv').config();

const app = express();
app.use(cors());
app.use(bodyParser.json());
const PORT = 5000;

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

/**
 * POST /login
 * Body: { username, password }
 * Success: { accessToken, refreshToken, role }
 */
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const foundUser = await Credential.findOne({ username, password });
  if (!foundUser) return res.status(401).send('Invalid credentials');

  const accessToken = jwt.sign(
    { username, role: foundUser.role },
    process.env.ACCESS_TOKEN_SECRET,
    { expiresIn: '30s' }
  );
  const refreshToken = jwt.sign(
```

```
    { username },
    process.env.REFRESH_TOKEN_SECRET,
    { expiresIn: '1d' }
  );

  foundUser.refresh_token = refreshToken;
  await foundUser.save();
  res.json({ accessToken, refreshToken, role: foundUser.role });
});

app.listen(PORT, () => console.log(`🔐 Auth server running on http://localhost:${PORT}`));
```

Student Service – studentserver

`studentserver/models/Student.js`

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

const StudentSchema = new mongoose.Schema({
  name: String,
  pic: String,
  email: String,
  batch: String,
  certificate: String,
  username: { type: String, unique: true },
  course_type: String,
  module: { type: String, default: 'A' }  // tracks current module for chat path
});

module.exports = mongoose.model('Student', StudentSchema);
```

`studentserver/models/Course.js`

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

const CourseSchema = new mongoose.Schema({
  course_type: { type: String, unique: true },
  modules: [String]
});

module.exports = mongoose.model('Course', CourseSchema);
```

studentserver/server.js

```
const express = require('express');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const mongoose = require('mongoose');
const Student = require('./models/Student');
const Course = require('./models/Course');
require('dotenv').config();

const app = express();
app.use(cors());
app.use(express.json());
const PORT = 5001;

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

function verifyToken(req, res, next) {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) return res.sendStatus(401);
  jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

/**
 * GET /student/me - profile of logged-in student
 */
app.get('/student/me', verifyToken, async (req, res) => {
  const student = await Student.findOne({ username: req.user.username });
  res.json(student);
});

/**
 * GET /student/modules - modules for the student's course
 */
app.get('/student/modules', verifyToken, async (req, res) => {
  const student = await Student.findOne({ username: req.user.username });
  const course = await Course.findOne({ course_type: student.course_type });
  res.json(course?.modules || []);
});
```

```
/**
 * POST /student/changepassword
 * Body: { username, oldPass, newPass }
 */
app.post('/student/changepassword', async (req, res) => {
  const { username, oldPass, newPass } = req.body;
  const cred = await mongoose.model('Credential', new mongoose.Schema({}),
'credentials')
                  .findOne({ username });
  if (!cred || cred.password !== oldPass) return res.status(400).send('Old
password incorrect');
  cred.password = newPass;
  await cred.save();
  res.send('Password updated');
});

app.listen(PORT, () => console.log(`🎓 Student server running on http://
localhost:${PORT}`));
```

Admin Service – adminserver

`adminserver/models/Admin.js`

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

const AdminSchema = new mongoose.Schema({
  name: String,
  username: { type: String, unique: true },
  email: String,
  pic: String,
  salary: Number,
  batches: [String]
});

module.exports = mongoose.model('Admin', AdminSchema);
```

`adminserver/server.js`

```
const express = require('express');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const mongoose = require('mongoose');
const Admin = require('./models/Admin');
```

```
require('dotenv').config();

const app = express();
app.use(cors());
app.use(express.json());
const PORT = 5002;

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

function verifyToken(req, res, next) {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) return res.sendStatus(401);
  jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

/**
 * GET /admin/me - profile of logged-in admin
 */
app.get('/admin/me', verifyToken, async (req, res) => {
  const admin = await Admin.findOne({ username: req.user.username });
  res.json(admin);
});

app.listen(PORT, () => console.log(`🛠️ Admin server running on http://
localhost:${PORT}`));
```

Realtime Chat Service – chatserver

`chatserver/server.js`

```
const express = require('express');
const http = require('http');
const socketIO = require('socket.io');
const fs = require('fs');
const path = require('path');
const cors = require('cors');
```

```
const app = express();
app.use(cors());
const server = http.createServer(app);
const io = socketIO(server, {
  cors: { origin: '*' }
});

const PORT = 5003;
const LOG_DIR = path.join(__dirname, 'chat_logs');
if (!fs.existsSync(LOG_DIR)) fs.mkdirSync(LOG_DIR);

function getLogPath(room) {
  const parts = room.split('/');
  const dir = path.join(LOG_DIR, ...parts.slice(0, -1));
  if (!fs.existsSync(dir)) fs.mkdirSync(dir, { recursive: true });
  return path.join(dir, parts.at(-1) + '.txt');
}
function loadChat(room) {
  const file = getLogPath(room);
  return fs.existsSync(file)
    ? fs.readFileSync(file, 'utf8').split('\n').filter(Boolean)
    : [];
}
function saveChat(room, message) {
  const file = getLogPath(room);
  fs.appendFileSync(file, message + '\n');
  fs.writeFileSync(file.replace('.txt', '_unread.flag'), '1');
}
function clearUnread(room) {
  const flag = getLogPath(room).replace('.txt', '_unread.flag');
  if (fs.existsSync(flag)) fs.unlinkSync(flag);
}
function hasUnread(room) {
  return fs.existsSync(getLogPath(room).replace('.txt', '_unread.flag'));
}

io.on('connection', socket => {
  socket.on('joinRoom', ({ name, room }) => {
    socket.join(room);
    socket.emit('chatHistory', loadChat(room));
    clearUnread(room);
    const joinMsg = `${name} joined ${room}`;
    saveChat(room, joinMsg);
    socket.to(room).emit('message', joinMsg);
  });

  socket.on('message', ({ name, room, message }) => {
    const msg = `${name}: ${message}`;
```

```
    saveChat(room, msg);
    io.to(room).emit('message', msg);
  });

  socket.on('checkUnread', ({ room }, cb) => cb(hasUnread(room)));
});

server.listen(PORT, () =>
  console.log(`📰 Chat server running on http://localhost:${PORT}`)
);
```

React Front-end – lms-client

`lms-client/package.json` ***(minimal)***

```
{
  "name": "lms-client",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "axios": "^1.6.2",
    "react": "^18.3.0",
    "react-dom": "^18.3.0",
    "react-router-dom": "^6.23.0",
    "socket.io-client": "^4.7.5"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

`src/context/UserContext.js`

```
import { createContext, useState } from 'react';

export const UserContext = createContext(null);

export function UserProvider({ children }) {
  const [user, setUser] = useState(null);
  return (
    <UserContext.Provider value={{ user, setUser }}>
      {children}
```

```
    </UserContext.Provider>
  );
}
```

---

src/pages/Login.js

```
import { useState, useContext } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { UserContext } from '../context/UserContext';

export default function Login() {
  const { setUser } = useContext(UserContext);
  const nav = useNavigate();
  const [form, setForm] = useState({ username: '', password: '' });
  const [error, setError] = useState('');

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

  async function handleSubmit(e) {
    e.preventDefault();
    try {
      const {
        data: { accessToken, refreshToken, role }
      } = await axios.post('http://localhost:5000/login', form);

      localStorage.setItem('accessToken', accessToken);
      localStorage.setItem('refreshToken', refreshToken);
      setUser({ username: form.username, role });
      role === 'student' ? nav('/studenthome') : nav('/adminhome');
    } catch (err) {
      setError(err.response?.data || 'Login failed');
    }
  }

  return (
    <form onSubmit={handleSubmit}>
      <input name="username" onChange={handleChange} placeholder="Username" />
      <input
        type="password"
        name="password"
        onChange={handleChange}
        placeholder="Password"
      />
```

```
      <button type="submit">Login</button>
      {error && <p style={{ color: 'red' }}>{error}</p>}
    </form>
  );
}
```

---

src/pages/StudentHome.js

```
import { useEffect, useState, useContext } from 'react';
import axios from 'axios';
import { UserContext } from '../context/UserContext';
import { useNavigate } from 'react-router-dom';

export default function StudentHome() {
  const { user } = useContext(UserContext);
  const nav = useNavigate();
  const [profile, setProfile] = useState(null);

  useEffect(() => {
    axios
      .get('http://localhost:5001/student/me', {
        headers: { Authorization: `Bearer ${localStorage.getItem('accessToken')}
` }
      })
      .then(res => setProfile(res.data));
  }, []);

  if (!profile) return <p>Loading…</p>;

  return (
    <div>
      <h1>Welcome {profile.name}</h1>
      <p>Batch: {profile.batch}</p>
      <button onClick={() => nav('/studentmodule')}>Modules</button>
      <button onClick={() => nav('/profile')}>Profile</button>
      <button onClick={() => nav('/chat')}>Chat</button>
    </div>
  );
}
```

src/pages/StudentModule.js

```
import { useEffect, useState } from 'react';
import axios from 'axios';
```

```
export default function StudentModule() {
  const [modules, setModules] = useState([]);

  useEffect(() => {
    axios
      .get('http://localhost:5001/student/modules', {
        headers: { Authorization: `Bearer ${localStorage.getItem('accessToken')}
` }
      })
      .then(res => setModules(res.data));
  }, []);

  return (
    <div>
      <h2>Modules</h2>
      <ul>{modules.map(m => <li key={m}>{m}</li>)}</ul>
    </div>
  );
}
```

src/pages/Profile.js

```
import { useEffect, useState } from 'react';
import axios from 'axios';

export default function Profile() {
  const [profile, setProfile] = useState(null);

  useEffect(() => {
    axios
      .get('http://localhost:5001/student/me', {
        headers: { Authorization: `Bearer ${localStorage.getItem('accessToken')}
` }
      })
      .then(res => setProfile(res.data));
  }, []);

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

  return (
    <div>
      <img src={profile.pic} alt="Profile" height={120} />
      <h2>{profile.name}</h2>
      <p>{profile.email}</p>
      <p>Batch: {profile.batch}</p>
```

```
      <p>Course: {profile.course_type}</p>
      <p>Certificate: {profile.certificate}</p>
    </div>
  );
}
```

`src/pages/Chat.js` ***(Student chat)***

```
import { useEffect, useState, useContext } from 'react';
import { io } from 'socket.io-client';
import { UserContext } from '../context/UserContext';

const socket = io('http://localhost:5003');

export default function Chat() {
  const { user } = useContext(UserContext);
  const [chat, setChat] = useState([]);
  const [message, setMessage] = useState('');
  const [mode, setMode] = useState('');
  const [room, setRoom] = useState('');

  useEffect(() => {
    socket.on('chatHistory', history => setChat(history));
    socket.on('message', msg => setChat(prev => [...prev, msg]));
  }, []);

  function joinRoom(selected) {
    let r = '';
    if (selected === 'group') r = `${user.batch}/${user.module}/group`;
    else r = `${user.name}/${user.module}`; // student-admin private room
    setRoom(r);
    socket.emit('joinRoom', { name: user.name, room: r });
    setMode(selected);
  }

  function sendMessage() {
    if (message && room) {
      socket.emit('message', { name: user.name, room, message });
      setMessage('');
    }
  }

  return (
    <div>
      <h2>Student Chat</h2>
```

```
      {!room && (
        <>
          <button onClick={() => joinRoom('group')}>Group Chat</button>
          <button onClick={() => joinRoom('admin')}>Admin Chat</button>
        </>
      )}
      {room && (
        <>
          <h4>{mode.toUpperCase()} CHAT</h4>
          <input
            value={message}
            onChange={e => setMessage(e.target.value)}
            placeholder="Message"
          />
          <button onClick={sendMessage}>Send</button>
          <div>
            {chat.map((m, i) => (
              <div key={i}>{m}</div>
            ))}
          </div>
        </>
      )}
    </div>
  );
}
```

---

src/pages/AdminHome.js

```
import { useEffect, useState, useContext } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { UserContext } from '../context/UserContext';

export default function AdminHome() {
  const [batches, setBatches] = useState([]);
  const { user } = useContext(UserContext);
  const nav = useNavigate();

  useEffect(() => {
    axios
      .get('http://localhost:5002/admin/me', {
        headers: { Authorization: `Bearer ${localStorage.getItem('accessToken')}
` }
      })
      .then(res => setBatches(res.data.batches));
```

```
  }, []);

  return (
    <div>
      <h1>Welcome {user?.name}</h1>
      <button onClick={() => nav('/adminprofile')}>Profile</button>
      {batches.map(b => (
        <button key={b} onClick={() => nav('/adminbatch')}>{b}</button>
      ))}
      <button onClick={() => nav('/adminchat')}>Chat</button>
    </div>
  );
}
```

src/pages/AdminProfile.js

```
import { useContext } from 'react';
import { UserContext } from '../context/UserContext';

export default function AdminProfile() {
  const { user } = useContext(UserContext);
  return (
    <div>
      <img src={user.pic} alt="Profile" height={120} />
      <h2>{user.name}</h2>
      <p>{user.email}</p>
      <p>Salary: ₹{user.salary}</p>
    </div>
  );
}
```

src/pages/AdminBatch.js

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

export default function AdminBatch() {
  const nav = useNavigate();
  return (
    <div>
      <h2>Admin Batch Page</h2>
      <button onClick={() => nav('/adminhome')}>Home</button>
      <button onClick={() => nav('/adminchat')}>Chat</button>
    </div>
```

```
  );
}
```

src/pages/AdminChat.js

```
import { useEffect, useState } from 'react';
import { io } from 'socket.io-client';

const socket = io('http://localhost:5003');

export default function AdminChat() {
  // In production, build this list from DB; hard-coded for demo
  const [rooms, setRooms] = useState(['TT1/C/group', 'John Doe/C']);
  const [selectedRoom, setSelectedRoom] = useState(null);
  const [chat, setChat] = useState([]);
  const [message, setMessage] = useState('');
  const [unread, setUnread] = useState({});

  useEffect(() => {
    // Check unread flags once on mount
    rooms.forEach(r => {
      socket.emit('checkUnread', { room: r }, has => {
        if (has) setUnread(prev => ({ ...prev, [r]: true }));
      });
    });
    socket.on('chatHistory', history => setChat(history));
    socket.on('message', msg => setChat(prev => [...prev, msg]));
  }, [rooms]);

  function joinRoom(room) {
    setSelectedRoom(room);
    socket.emit('joinRoom', { name: 'Admin', room });
    setUnread(prev => ({ ...prev, [room]: false }));
  }

  function sendMessage() {
    if (message) {
      socket.emit('message', { name: 'Admin', room: selectedRoom, message });
      setMessage('');
    }
  }

  return (
    <div>
      <h2>Admin Chat</h2>
      <div>
```

```
        {rooms.map(r => (
          <button
            key={r}
            onClick={() => joinRoom(r)}
            style={{ border: unread[r] ? '2px solid blue' : '1px solid gray' }}
          >
            {r}
          </button>
        ))}
      </div>
      {selectedRoom && (
        <>
          <input
            value={message}
            onChange={e => setMessage(e.target.value)}
            placeholder="Message"
          />
          <button onClick={sendMessage}>Send</button>
          <div>
            {chat.map((m, i) => (
              <div key={i}>{m}</div>
            ))}
          </div>
        </>
      )}
    </div>
  );
}
```

Quick start

1. **Install MongoDB** and run it locally on the default port.
2. From the repo root, copy `shared.env` into each server folder as `.env`.
3. Install dependencies and start each micro-service in a separate terminal:

```
# Auth
cd loginserver && npm i && node server.js

# Student
cd ../studentserver && npm i && node server.js

# Admin
cd ../adminserver && npm i && node server.js
```

```
# Chat (Socket.IO)
cd ../chatserver && npm i && node server.js
```

4. Finally start the React client:

```
cd ../lms-client && npm i && npm start
```

The system now supports **student** and **admin** roles with JWT-based authentication, dedicated REST APIs, and real-time chat rooms.