controllers---AlumniController.js

```
const Alumni = require("../models/AlumniModel");
const Student = require("../models/StudentModel"); // Import the Student model

// 🟢 Fetch Alumni Profile
const getAlumniProfile = async (req, res) => {
  try {
    // Find alumni by userId (assuming req.user.id is correct)
    const alumni = await Alumni.findById(req.user.id);

    if (!alumni) {
      return res.status(404).json({ message: "Alumni profile not found" });
    }

    res.json(alumni);  // Send the alumni profile as JSON
  } catch (error) {
    console.error("Error fetching alumni profile:", error);
    res.status(500).json({ message: "Server error", error: error.message });
  }
};

// 🟢 Update Alumni Profile
const updateAlumniProfile = async (req, res) => {
  try {
    // Find alumni by userId (assuming req.user.id is correct)
    const alumni = await Alumni.findById(req.user.id);

    if (!alumni) {
      return res.status(404).json({ message: "Alumni profile not found" });
    }

    // Update the alumni profile with the new data
    Object.assign(alumni, req.body);

    // Save the updated alumni profile to the database
    await alumni.save();

    res.json({ message: "Profile updated successfully", alumni });
  } catch (error) {
    console.error("Error updating alumni profile:", error);
    res.status(500).json({ message: "Server error", error: error.message });
  }
};

// 🟢 Fetch Student Profile
const getStudentProfile = async (req, res) => {
  try {
    // Find student by userId (assuming req.user.id is correct)
    const student = await Student.findById(req.user.id);

    if (!student) {
```

```
        return res.status(404).json({ message: "Student profile not found" });
      }

      res.json(student);  // Send the student profile as JSON
    } catch (error) {
      console.error("Error fetching student profile:", error);
      res.status(500).json({ message: "Server error", error: error.message });
    }
};

// 🟢 Update Student Profile
const updateStudentProfile = async (req, res) => {
    try {
      // Find student by userId (assuming req.user.id is correct)
      const student = await Student.findById(req.user.id);

      if (!student) {
        return res.status(404).json({ message: "Student profile not found" });
      }

      // Update the student profile with the new data
      Object.assign(student, req.body);

      // Save the updated student profile to the database
      await student.save();

      res.json({ message: "Profile updated successfully", student });
    } catch (error) {
      console.error("Error updating student profile:", error);
      res.status(500).json({ message: "Server error", error: error.message });
    }
};

module.exports = {
    getAlumniProfile,
    updateAlumniProfile,
    getStudentProfile,  // Export the new function for getting student profile
    updateStudentProfile // Export the new function for updating student profile
};
```

middleware---authMiddleware.js

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

const authMiddleware = (req, res, next) => {
  try {
    // Log the Authorization header (only in development mode)
    if (process.env.NODE_ENV === "development") {
      console.log("Authorization Header:", req.headers.authorization);
    }

    // Ensure the Authorization header exists and follows "Bearer <token>" format
    if (!req.headers.authorization || !req.headers.authorization.startsWith("Bearer ")) {
      return res.status(401).json({ message: "Authorization token is missing or malformed" });
    }

    // Extract the token from the "Bearer <token>" format
    const token = req.headers.authorization.split(" ")[1];

    // Ensure JWT_SECRET is set
    if (!process.env.JWT_SECRET) {
      console.error("JWT_SECRET is missing in environment variables.");
      return res.status(500).json({ message: "Server configuration error" });
    }

    // Verify token
    const decoded = jwt.verify(token, process.env.JWT_SECRET);

    // Log decoded token (only in development mode)
    if (process.env.NODE_ENV === "development") {
      console.log("Decoded Token:", decoded);
    }

    // Attach user data to request object
    req.user = decoded;
    next();
  } catch (error) {
    console.error("Token Verification Error:", error);

    if (error.name === "TokenExpiredError") {
      return res.status(401).json({ message: "Token has expired, please login again" });
    } else if (error.name === "JsonWebTokenError") {
      return res.status(403).json({ message: "Invalid token" });
    }

    return res.status(500).json({ message: "Internal server error" });
  }
};
```

```
module.exports = authMiddleware;
```

server.js

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

// Import Routes
const uploadRoutes = require("./routes/upload");
const registerRoute = require("./routes/register");
const loginRoute = require("./routes/login");
const resourceRoutes = require("./routes/resources");
const downloadRoutes = require("./routes/download");
const reviewRoutes = require("./routes/reviewRoutes");
const alumniRoutes = require("./routes/alumniRoutes");  // ✅ Add this
const studentRoutes = require("./routes/studentRoutes");

const app = express();

// Middleware
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cors({ origin: "http://localhost:3000", credentials: true }));

// ✅ Serve uploaded files statically
app.use("/uploads", express.static(path.join(__dirname, "uploads")));

// ✅ Use Routes
app.use("/api/upload", uploadRoutes);
app.use("/api", registerRoute);
app.use("/api", loginRoute);
app.use("/api/resources", resourceRoutes);
app.use("/", downloadRoutes);
app.use("/api/reviews", reviewRoutes);
app.use("/api/alumni", alumniRoutes);  // ✅ Ensure this line is added
app.use("/api/student", studentRoutes);

// ✅ Ensure uploads/ folder exists
const fs = require("fs");
const uploadDir = path.join(__dirname, "uploads");
if (!fs.existsSync(uploadDir)) {
   fs.mkdirSync(uploadDir, { recursive: true });
   console.log("✅ Created 'uploads' folder");
```

```
}

// ✅ Connect to MongoDB
mongoose.connect("mongodb://127.0.0.1:27017/alumnilink", {})
.then(() => console.log("✅ MongoDB Connected"))
.catch(err => console.error("❌ MongoDB connection error:", err));

// ✅ Start Server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`)); // ✅ Fix string interpolation
```