Below is a **complete working starter project** (HTML + CSS + Bootstrap frontend + Node/Express + MongoDB backend) for your **Internship & Project/Research Submission Portal**.

You will get:

- ✅ Installation steps
- ✅ Folder structure
- ✅ ALL code files (copy–paste)
- ✅ Student upload + Faculty review + Admin assign
- ✅ PDF upload using Multer
- ✅ JWT login + bcrypt password hashing

---

1) Software / Installation Needed

Install these (one time)

1. **Node.js (LTS)**
2. **MongoDB Community Server** (or MongoDB Compass + Local MongoDB)
3. **VS Code** (recommended)

Check versions

Open terminal / cmd:

node -v

npm -v

mongod --version

---

2) Project Setup (Step-by-step)

Step A: Create folder

mkdir submission-portal

cd submission-portal

Step B: Create backend folder

mkdir backend

cd backend

npm init -y

Step C: Install packages

npm i express mongoose bcrypt jsonwebtoken cors multer dotenv

npm i nodemon --save-dev

Step D: Add start script

Open backend/package.json and add:

```
"scripts": {
  "start": "node server.js",
  "dev": "nodemon server.js"
}
```

Step E: Start MongoDB

- If MongoDB is installed as service, it runs automatically.
- Otherwise run:

```
mongod
```

Step F: Run backend

```
npm run dev
```

Open in browser:

- Backend health: http://localhost:5000

---

3) Folder Structure (Create exactly like this)

```
submission-portal/
  backend/
    server.js
    .env
    package.json
    models/
      User.js
      Submission.js
      Review.js
    middleware/
      auth.js
    routes/
      authRoutes.js
      submissionRoutes.js
```

adminRoutes.js

facultyRoutes.js

uploads/ (auto created)

public/

index.html

register.html

student.html

faculty.html

admin.html

---

4) Backend Code (Node + Express + MongoDB)

(1) backend/.env

```
PORT=5000
MONGO_URI=mongodb://127.0.0.1:27017/submission_portal
JWT_SECRET=super_secret_key_change_this
```

---

(2) backend/server.js

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

const authRoutes = require("./routes/authRoutes");
const submissionRoutes = require("./routes/submissionRoutes");
const adminRoutes = require("./routes/adminRoutes");
const facultyRoutes = require("./routes/facultyRoutes");

const app = express();

app.use(cors());
```

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

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

// Serve frontend (HTML pages)
app.use(express.static(path.join(__dirname, "public")));

// API routes
app.use("/api/auth", authRoutes);
app.use("/api/submissions", submissionRoutes);
app.use("/api/admin", adminRoutes);
app.use("/api/faculty", facultyRoutes);

app.get("/health", (req, res) => res.json({ ok: true, message: "Server running" }));

mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    console.log("MongoDB connected");
    app.listen(process.env.PORT, () => console.log(`Server started on port ${process.env.PORT}`));
  })
  .catch((err) => console.error("Mongo error:", err.message));
```

---

(3) backend/models/User.js

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

const userSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
```

```
    email: { type: String, required: true, unique: true, lowercase: true },
    passwordHash: { type: String, required: true },
    role: { type: String, enum: ["student", "faculty", "admin"], required: true },
    dept: { type: String, default: "" },
    year: { type: String, default: "" }
  },
  { timestamps: true }
);

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

---

(4) backend/models/Submission.js

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

const submissionSchema = new mongoose.Schema(
  {
    studentId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
    title: { type: String, required: true },
    type: { type: String, enum: ["internship", "project", "research"], required: true },
    domain: { type: String, default: "" },
    companyOrGuide: { type: String, default: "" },
    filePath: { type: String, required: true },
    status: { type: String, enum: ["Submitted", "Assigned", "Reviewed", "Approved", "Resubmission
Required"], default: "Submitted" },
    assignedFacultyId: { type: mongoose.Schema.Types.ObjectId, ref: "User", default: null },
    version: { type: Number, default: 1 }
  },
  { timestamps: true }
);

module.exports = mongoose.model("Submission", submissionSchema);
```

---

(5) backend/models/Review.js

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

const reviewSchema = new mongoose.Schema(
 {
  submissionId: { type: mongoose.Schema.Types.ObjectId, ref: "Submission", required: true },
  facultyId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  remarks: { type: String, default: "" },
  marks: { type: Number, default: 0 },
  decision: { type: String, enum: ["Approved", "Resubmission Required"], required: true }
 },
 { timestamps: true }
);

module.exports = mongoose.model("Review", reviewSchema);
```

---

(6) backend/middleware/auth.js

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

function auth(requiredRole = null) {
 return (req, res, next) => {
  try {
   const header = req.headers.authorization || "";
   const token = header.startsWith("Bearer ") ? header.slice(7) : null;
   if (!token) return res.status(401).json({ message: "No token" });

   const payload = jwt.verify(token, process.env.JWT_SECRET);
   req.user = payload;

   if (requiredRole && payload.role !== requiredRole) {
```

```
      return res.status(403).json({ message: "Forbidden (role mismatch)" });
    }
    next();
  } catch (err) {
    return res.status(401).json({ message: "Invalid token" });
  }
 };
}

module.exports = auth;
```

---

(7) backend/routes/authRoutes.js

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

// Register (Student by default, but allow role if you want for testing)
router.post("/register", async (req, res) => {
 try {
  const { name, email, password, role = "student", dept = "", year = "" } = req.body;

  if (!name || !email || !password) return res.status(400).json({ message: "Missing fields" });

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

  const passwordHash = await bcrypt.hash(password, 10);

  const user = await User.create({ name, email, passwordHash, role, dept, year });
  return res.json({ message: "Registered", userId: user._id });
```

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

router.post("/login", async (req, res) => {
  try {
    const { email, password } = req.body;

    const user = await User.findOne({ email });
    if (!user) return res.status(401).json({ message: "Invalid credentials" });

    const ok = await bcrypt.compare(password, user.passwordHash);
    if (!ok) return res.status(401).json({ message: "Invalid credentials" });

    const token = jwt.sign(
      { userId: user._id.toString(), role: user.role, name: user.name },
      process.env.JWT_SECRET,
      { expiresIn: "1d" }
    );

    return res.json({ token, role: user.role, name: user.name });
  } catch (err) {
    return res.status(500).json({ message: err.message });
  }
});

module.exports = router;
```

---

(8) backend/routes/submissionRoutes.js

```
const router = require("express").Router();
```

```
const path = require("path");
const fs = require("fs");
const multer = require("multer");

const auth = require("../middleware/auth");
const Submission = require("../models/Submission");
const Review = require("../models/Review");

// Ensure uploads folder exists
const uploadDir = path.join(__dirname, "..", "uploads");
if (!fs.existsSync(uploadDir)) fs.mkdirSync(uploadDir);

const storage = multer.diskStorage({
  destination: (req, file, cb) => cb(null, uploadDir),
  filename: (req, file, cb) => {
    const safeName = Date.now() + "-" + file.originalname.replace(/\s+/g, "_");
    cb(null, safeName);
  }
});

function pdfOnly(req, file, cb) {
  if (file.mimetype === "application/pdf") cb(null, true);
  else cb(new Error("Only PDF allowed"));
}

const upload = multer({ storage, fileFilter: pdfOnly, limits: { fileSize: 10 * 1024 * 1024 } });

// Student upload
router.post("/upload", auth("student"), upload.single("report"), async (req, res) => {
  try {
    const { title, type, domain, companyOrGuide } = req.body;
```

```
    if (!req.file) return res.status(400).json({ message: "PDF required" });
    if (!title || !type) return res.status(400).json({ message: "Title and type required" });

    const filePath = `/uploads/${req.file.filename}`;

    const submission = await Submission.create({
      studentId: req.user.userId,
      title,
      type,
      domain: domain || "",
      companyOrGuide: companyOrGuide || "",
      filePath,
      status: "Submitted"
    });

    return res.json({ message: "Uploaded", submission });
  } catch (err) {
    return res.status(500).json({ message: err.message });
  }
});

// Student view own submissions + reviews
router.get("/mine", auth("student"), async (req, res) => {
  const submissions = await Submission.find({ studentId: req.user.userId })
    .populate("assignedFacultyId", "name email")
    .sort({ createdAt: -1 });

  const ids = submissions.map(s => s._id);
  const reviews = await Review.find({ submissionId: { $in: ids } }).populate("facultyId", "name email");

  return res.json({ submissions, reviews });
```

```
});

module.exports = router;
```

---

(9) backend/routes/adminRoutes.js

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

// Get all students/faculty
router.get("/users", auth("admin"), async (req, res) => {
  const users = await User.find({}, "name email role dept year").sort({ createdAt: -1 });
  res.json({ users });
});

// Get all submissions
router.get("/submissions", auth("admin"), async (req, res) => {
  const submissions = await Submission.find({})
    .populate("studentId", "name email dept year")
    .populate("assignedFacultyId", "name email")
    .sort({ createdAt: -1 });
  res.json({ submissions });
});

// Assign faculty to a submission
router.post("/assign", auth("admin"), async (req, res) => {
  const { submissionId, facultyId } = req.body;
  if (!submissionId || !facultyId) return res.status(400).json({ message: "Missing fields" });

  const sub = await Submission.findByIdAndUpdate(
```

```
    submissionId,
    { assignedFacultyId: facultyId, status: "Assigned" },
    { new: true }
  );
  res.json({ message: "Assigned", submission: sub });
});

module.exports = router;
```

---

(10) backend/routes/facultyRoutes.js

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

// Faculty: view assigned submissions
router.get("/assigned", auth("faculty"), async (req, res) => {
  const list = await Submission.find({ assignedFacultyId: req.user.userId })
    .populate("studentId", "name email dept year")
    .sort({ createdAt: -1 });

  res.json({ submissions: list });
});

// Faculty: evaluate (review)
router.post("/review", auth("faculty"), async (req, res) => {
  const { submissionId, remarks, marks, decision } = req.body;
  if (!submissionId || !decision) return res.status(400).json({ message: "Missing fields" });

  const sub = await Submission.findById(submissionId);
  if (!sub) return res.status(404).json({ message: "Submission not found" });
```

```
  if (String(sub.assignedFacultyId) !== req.user.userId) {
    return res.status(403).json({ message: "Not assigned to you" });
  }

  const review = await Review.create({
    submissionId,
    facultyId: req.user.userId,
    remarks: remarks || "",
    marks: Number(marks || 0),
    decision
  });

  sub.status = decision;
  await sub.save();

  res.json({ message: "Reviewed", review, updatedStatus: sub.status });
});

module.exports = router;
```

---

5) Frontend Code (HTML + Bootstrap)

All frontend files go in: backend/public/

---

(1) backend/public/index.html (Login)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Submission Portal - Login</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
</head>
<body class="bg-light">
<div class="container mt-5" style="max-width: 520px;">
  <div class="card shadow">
    <div class="card-body">
      <h3 class="mb-3">Login</h3>

      <div class="mb-2">
        <label>Email</label>
        <input id="email" class="form-control" type="email" placeholder="Enter email">
      </div>

      <div class="mb-3">
        <label>Password</label>
        <input id="password" class="form-control" type="password" placeholder="Enter password">
      </div>

      <button class="btn btn-primary w-100" onclick="login()">Login</button>

      <div class="mt-3 d-flex justify-content-between">
        <a href="register.html">Create Account</a>
        <span id="msg" class="text-danger"></span>
      </div>

      <hr>
      <small class="text-muted">
        Tip: Create one Admin and Faculty using Register page by selecting role (for testing).
      </small>
    </div>
  </div>
</div>
```

```
<script>
const API = "http://localhost:5000";

async function login() {
  const email = document.getElementById("email").value.trim();
  const password = document.getElementById("password").value.trim();
  const msg = document.getElementById("msg");
  msg.textContent = "";

  const res = await fetch(API + "/api/auth/login", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({ email, password })
  });

  const data = await res.json();
  if (!res.ok) {
    msg.textContent = data.message || "Login failed";
    return;
  }

  localStorage.setItem("token", data.token);
  localStorage.setItem("role", data.role);
  localStorage.setItem("name", data.name);

  if (data.role === "student") location.href = "student.html";
  else if (data.role === "faculty") location.href = "faculty.html";
  else location.href = "admin.html";
}
</script>
```

```
</body>

</html>
```

---

(2) backend/public/register.html

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="utf-8" />

  <title>Register</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">

</head>

<body class="bg-light">

<div class="container mt-5" style="max-width: 620px;">

  <div class="card shadow">

    <div class="card-body">

      <h3 class="mb-3">Register</h3>


      <div class="row g-2">

        <div class="col-md-6">

          <label>Name</label>

          <input id="name" class="form-control" placeholder="Full name">

        </div>

        <div class="col-md-6">

          <label>Email</label>

          <input id="email" class="form-control" placeholder="Email">

        </div>


        <div class="col-md-6">

          <label>Password</label>

          <input id="password" type="password" class="form-control" placeholder="Password">
```

```
      </div>

      <div class="col-md-6">
        <label>Role (for testing)</label>
        <select id="role" class="form-select">
          <option value="student">Student</option>
          <option value="faculty">Faculty</option>
          <option value="admin">Admin</option>
        </select>
      </div>

      <div class="col-md-6">
        <label>Department</label>
        <input id="dept" class="form-control" placeholder="e.g. CS">
      </div>
      <div class="col-md-6">
        <label>Year</label>
        <input id="year" class="form-control" placeholder="e.g. MSc-II">
      </div>
    </div>

    <button class="btn btn-success w-100 mt-3" onclick="register()">Create Account</button>
    <div class="mt-3 d-flex justify-content-between">
      <a href="index.html">Back to Login</a>
      <span id="msg" class="text-danger"></span>
    </div>
  </div>
 </div>
</div>

<script>
```

```
const API = "http://localhost:5000";

async function register() {
  const body = {
    name: document.getElementById("name").value.trim(),
    email: document.getElementById("email").value.trim(),
    password: document.getElementById("password").value.trim(),
    role: document.getElementById("role").value,
    dept: document.getElementById("dept").value.trim(),
    year: document.getElementById("year").value.trim()
  };

  const msg = document.getElementById("msg");
  msg.textContent = "";

  const res = await fetch(API + "/api/auth/register", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify(body)
  });

  const data = await res.json();
  if (!res.ok) { msg.textContent = data.message || "Register failed"; return; }

  alert("Registered successfully! Now login.");
  location.href = "index.html";
}
</script>
</body>
</html>
```

---

(3) backend/public/student.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Student Dashboard</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
<nav class="navbar navbar-dark bg-dark px-3">
  <span class="navbar-brand">Student Dashboard</span>
  <div class="text-white">
    <span id="who"></span>
    <button class="btn btn-sm btn-outline-light ms-2" onclick="logout()">Logout</button>
  </div>
</nav>

<div class="container mt-4">
  <div class="row g-3">
    <div class="col-md-5">
      <div class="card shadow">
        <div class="card-body">
          <h5>Upload Report (PDF)</h5>
          <div class="mb-2">
            <label>Title</label>
            <input id="title" class="form-control" placeholder="Project/Internship title">
          </div>
          <div class="mb-2">
            <label>Type</label>
            <select id="type" class="form-select">
```

```
          <option value="internship">Internship</option>
          <option value="project">Final Year Project</option>
          <option value="research">Research</option>
        </select>
      </div>
      <div class="mb-2">
        <label>Domain</label>
        <input id="domain" class="form-control" placeholder="AI / Web / Data Science">
      </div>
      <div class="mb-2">
        <label>Company / Guide</label>
        <input id="companyOrGuide" class="form-control" placeholder="Company or Guide name">
      </div>
      <div class="mb-3">
        <label>PDF File</label>
        <input id="pdf" type="file" accept="application/pdf" class="form-control">
      </div>
      <button class="btn btn-primary w-100" onclick="upload()">Upload</button>
      <div class="mt-2 text-danger" id="msg"></div>
    </div>
  </div>
</div>

<div class="col-md-7">
  <div class="card shadow">
    <div class="card-body">
      <h5>My Submissions</h5>
      <div class="table-responsive">
        <table class="table table-bordered table-sm">
          <thead class="table-dark">
            <tr>
```

```
            <th>Title</th><th>Type</th><th>Status</th><th>Faculty</th><th>File</th>
          </tr>
        </thead>
        <tbody id="rows"></tbody>
      </table>
    </div>
    <h6 class="mt-3">My Reviews</h6>
    <div id="reviews"></div>
  </div>
 </div>
</div>
</div>

<script>
const API = "http://localhost:5000";
const token = localStorage.getItem("token");
if (!token || localStorage.getItem("role") !== "student") location.href = "index.html";

document.getElementById("who").textContent = localStorage.getItem("name") || "Student";

function logout() {
 localStorage.clear();
 location.href = "index.html";
}

async function upload() {
 const msg = document.getElementById("msg");
 msg.textContent = "";

 const f = document.getElementById("pdf").files[0];
```

```
  if (!f) { msg.textContent = "Please select PDF"; return; }

  const fd = new FormData();
  fd.append("title", document.getElementById("title").value.trim());
  fd.append("type", document.getElementById("type").value);
  fd.append("domain", document.getElementById("domain").value.trim());
  fd.append("companyOrGuide", document.getElementById("companyOrGuide").value.trim());
  fd.append("report", f);

  const res = await fetch(API + "/api/submissions/upload", {
    method: "POST",
    headers: { "Authorization": "Bearer " + token },
    body: fd
  });

  const data = await res.json();
  if (!res.ok) { msg.textContent = data.message || "Upload failed"; return; }

  alert("Uploaded!");
  loadMine();
}

async function loadMine() {
  const res = await fetch(API + "/api/submissions/mine", {
    headers: { "Authorization": "Bearer " + token }
  });
  const data = await res.json();

  const tbody = document.getElementById("rows");
  tbody.innerHTML = "";
```

```
  (data.submissions || []).forEach(s => {
    const faculty = s.assignedFacultyId ? s.assignedFacultyId.name : "-";
    tbody.innerHTML += `
      <tr>
        <td>${s.title}</td>
        <td>${s.type}</td>
        <td>${s.status}</td>
        <td>${faculty}</td>
        <td><a target="_blank" href="${API + s.filePath}">Open PDF</a></td>
      </tr>`;
  });

  const revDiv = document.getElementById("reviews");
  revDiv.innerHTML = "";
  (data.reviews || []).forEach(r => {
    revDiv.innerHTML += `
      <div class="border rounded p-2 mb-2 bg-white">
        <b>Decision:</b> ${r.decision} | <b>Marks:</b> ${r.marks}<br/>
        <b>Remarks:</b> ${r.remarks}<br/>
        <small class="text-muted">Reviewed by: ${r.facultyId?.name || ""}</small>
      </div>`;
  });
}

loadMine();
</script>
</body>
</html>
```

---

(4) backend/public/faculty.html

```
<!DOCTYPE html>
```

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Faculty Dashboard</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
<nav class="navbar navbar-dark bg-dark px-3">
  <span class="navbar-brand">Faculty Dashboard</span>
  <div class="text-white">
    <span id="who"></span>
    <button class="btn btn-sm btn-outline-light ms-2" onclick="logout()">Logout</button>
  </div>
</nav>

<div class="container mt-4">
  <div class="card shadow">
    <div class="card-body">
      <h5>Assigned Submissions</h5>
      <div class="table-responsive">
        <table class="table table-bordered table-sm">
          <thead class="table-dark">
            <tr>
              <th>Student</th><th>Title</th><th>Status</th><th>PDF</th><th>Action</th>
            </tr>
          </thead>
          <tbody id="rows"></tbody>
        </table>
      </div>
    </div>
```

```
  </div>
</div>

<!-- Review Modal (simple) -->
<div class="modal fade" id="reviewModal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Evaluate Submission</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
      </div>
      <div class="modal-body">
        <input type="hidden" id="sid">
        <div class="mb-2">
          <label>Marks</label>
          <input id="marks" class="form-control" type="number" value="0">
        </div>
        <div class="mb-2">
          <label>Decision</label>
          <select id="decision" class="form-select">
            <option value="Approved">Approved</option>
            <option value="Resubmission Required">Resubmission Required</option>
          </select>
        </div>
        <div class="mb-2">
          <label>Remarks</label>
          <textarea id="remarks" class="form-control" rows="3"></textarea>
        </div>
        <div class="text-danger" id="msg"></div>
      </div>
      <div class="modal-footer">
```

```
      <button class="btn btn-secondary" data-bs-dismiss="modal">Cancel</button>
      <button class="btn btn-primary" onclick="submitReview()">Submit</button>
    </div>
   </div>
  </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script>
const API = "http://localhost:5000";
const token = localStorage.getItem("token");
if (!token || localStorage.getItem("role") !== "faculty") location.href = "index.html";

document.getElementById("who").textContent = localStorage.getItem("name") || "Faculty";
const modal = new bootstrap.Modal(document.getElementById('reviewModal'));

function logout(){ localStorage.clear(); location.href="index.html"; }

async function loadAssigned() {
  const res = await fetch(API + "/api/faculty/assigned", {
    headers: { "Authorization": "Bearer " + token }
  });
  const data = await res.json();

  const tbody = document.getElementById("rows");
  tbody.innerHTML = "";

  (data.submissions || []).forEach(s => {
    tbody.innerHTML += `
      <tr>
```

```
      <td>${s.studentId?.name || ""}</td>
      <td>${s.title}</td>
      <td>${s.status}</td>
      <td><a target="_blank" href="${API + s.filePath}">Open PDF</a></td>
      <td><button class="btn btn-sm btn-success"
onclick="openReview('${s._id}')">Review</button></td>
    </tr>`;
  });
}

function openReview(id) {
  document.getElementById("sid").value = id;
  document.getElementById("msg").textContent = "";
  document.getElementById("marks").value = 0;
  document.getElementById("remarks").value = "";
  modal.show();
}

async function submitReview() {
  const msg = document.getElementById("msg");
  msg.textContent = "";

  const body = {
    submissionId: document.getElementById("sid").value,
    marks: document.getElementById("marks").value,
    decision: document.getElementById("decision").value,
    remarks: document.getElementById("remarks").value.trim()
  };

  const res = await fetch(API + "/api/faculty/review", {
    method: "POST",
```

```
    headers: {
      "Content-Type": "application/json",
      "Authorization": "Bearer " + token
    },
    body: JSON.stringify(body)
  });

  const data = await res.json();
  if (!res.ok) { msg.textContent = data.message || "Review failed"; return; }

  modal.hide();
  loadAssigned();
}

loadAssigned();
</script>
</body>
</html>
```

---

(5) backend/public/admin.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Admin Panel</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
<nav class="navbar navbar-dark bg-dark px-3">
  <span class="navbar-brand">Admin Panel</span>
```

```
  <div class="text-white">
    <span id="who"></span>
    <button class="btn btn-sm btn-outline-light ms-2" onclick="logout()">Logout</button>
  </div>
</nav>

<div class="container mt-4">
  <div class="row g-3">
    <div class="col-md-5">
      <div class="card shadow">
        <div class="card-body">
          <h5>Users</h5>
          <div class="small text-muted mb-2">Use this to get Faculty ID for assigning.</div>
          <div class="table-responsive" style="max-height:420px; overflow:auto;">
            <table class="table table-bordered table-sm">
              <thead class="table-dark">
                <tr><th>Name</th><th>Role</th><th>Email</th></tr>
              </thead>
              <tbody id="users"></tbody>
            </table>
          </div>
        </div>
      </div>
    </div>

    <div class="col-md-7">
      <div class="card shadow">
        <div class="card-body">
          <h5>All Submissions</h5>
          <div class="table-responsive">
            <table class="table table-bordered table-sm">
```

```
          <thead class="table-dark">
           <tr>
             <th>Student</th><th>Title</th><th>Status</th><th>Assigned
Faculty</th><th>PDF</th><th>Assign</th>
           </tr>
          </thead>
          <tbody id="subs"></tbody>
         </table>
        </div>

        <div class="mt-3 p-2 border rounded bg-white">
         <h6>Assign Faculty</h6>
         <div class="row g-2">
          <div class="col-md-6">
           <input id="submissionId" class="form-control" placeholder="Paste Submission ID">
          </div>
          <div class="col-md-6">
           <input id="facultyId" class="form-control" placeholder="Paste Faculty ID">
          </div>
         </div>
         <button class="btn btn-primary mt-2" onclick="assign()">Assign</button>
         <span id="msg" class="text-danger ms-2"></span>
        </div>
       </div>
     </div>
   </div>
  </div>
</div>

<script>
const API = "http://localhost:5000";
```

```
const token = localStorage.getItem("token");
if (!token || localStorage.getItem("role") !== "admin") location.href = "index.html";
document.getElementById("who").textContent = localStorage.getItem("name") || "Admin";

function logout(){ localStorage.clear(); location.href="index.html"; }

async function loadUsers() {
  const res = await fetch(API + "/api/admin/users", {
    headers: { "Authorization": "Bearer " + token }
  });
  const data = await res.json();
  const tbody = document.getElementById("users");
  tbody.innerHTML = "";

  (data.users || []).forEach(u => {
    tbody.innerHTML += `
      <tr title="Copy ID from console if needed">
        <td>${u.name}</td><td>${u.role}</td><td>${u.email}</td>
      </tr>`;
  });

  // Also log full users with IDs in console (easy copy)
  console.log("USERS (with IDs):", data.users);
}

async function loadSubs() {
  const res = await fetch(API + "/api/admin/submissions", {
    headers: { "Authorization": "Bearer " + token }
  });
  const data = await res.json();
  const tbody = document.getElementById("subs");
```

```
  tbody.innerHTML = "";

  (data.submissions || []).forEach(s => {
    const student = s.studentId?.name || "";
    const faculty = s.assignedFacultyId?.name || "-";
    tbody.innerHTML += `
      <tr>
        <td>${student}</td>
        <td>${s.title}</td>
        <td>${s.status}</td>
        <td>${faculty}</td>
        <td><a target="_blank" href="${API + s.filePath}">Open</a></td>
        <td>
          <button class="btn btn-sm btn-warning"
            onclick="prefill('${s._id}')">Use</button>
        </td>
      </tr>`;
  });

  console.log("SUBMISSIONS (with IDs):", data.submissions);
}

function prefill(id) {
  document.getElementById("submissionId").value = id;
}

async function assign() {
  const msg = document.getElementById("msg");
  msg.textContent = "";

  const body = {
```

```
    submissionId: document.getElementById("submissionId").value.trim(),
    facultyId: document.getElementById("facultyId").value.trim()
  };

  const res = await fetch(API + "/api/admin/assign", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": "Bearer " + token
    },
    body: JSON.stringify(body)
  });

  const data = await res.json();
  if (!res.ok) { msg.textContent = data.message || "Assign failed"; return; }

  alert("Assigned!");
  loadSubs();
}

loadUsers();
loadSubs();
</script>
</body>
</html>
```

---

6) How to Test (Working Demo)

1) Run backend

cd submission-portal/backend

npm run dev

2) Open browser

Go to:

- http://localhost:5000/index.html

3) Register users (for testing)

- Create **1 Admin**
- Create **1 Faculty**
- Create **1 Student**

4) Student uploads PDF

- Login as Student → Upload PDF

5) Admin assigns faculty

- Login as Admin → Open console (F12) to copy Faculty ID
- Assign faculty to that submission

6) Faculty reviews

- Login as Faculty → Review → Approve or Resubmission

7) Student sees feedback

- Login again as Student → view "My Reviews"

If you want, I can also add (extra features)

- ✅ Resubmission as version 2, 3 (proper versioning)
- ✅ Admin deadline setting
- ✅ Email notification (optional)
- ✅ Better UI (sidebar layout with Bootstrap)

Just say: **"add version resubmission + deadlines"** and I'll extend this code.