# OralVis Healthcare — Full-Stack Project

This document contains a ready-to-upload Git repository structure, file-by-file source code, and deployment + run instructions for the OralVis Healthcare assignment (Technician upload, Dentist viewer, PDF export).

## Repo file tree

oralvis-healthcare/  
├─ backend/  
│ ├─ package.json  
│ ├─ .env.example  
│ ├─ server.js  
│ ├─ db.js  
│ ├─ routes/  
│ │ ├─ auth.js  
│ │ └─ scans.js  
│ ├─ middleware/  
│ │ └─ auth.js  
│ └─ utils/  
│ └─ cloudinary.js  
├─ frontend/  
│ ├─ package.json  
│ ├─ vite.config.js  
│ ├─ .env.example  
│ └─ src/  
│ ├─ main.jsx  
│ ├─ App.jsx  
│ ├─ api.js  
│ ├─ pages/  
│ │ ├─ Login.jsx  
│ │ ├─ TechnicianUpload.jsx  
│ │ └─ DentistViewer.jsx  
│ └─ components/  
│ └─ ProtectedRoute.jsx  
├─ README.md  
└─ .gitignore

## Backend — backend/package.json

{  
 "name": "oralvis-backend",  
 "version": "1.0.0",  
 "main": "server.js",  
 "scripts": {  
 "start": "node server.js",  
 "dev": "nodemon server.js"  
 },  
 "dependencies": {  
 "bcrypt": "^5.1.0",  
 "cloudinary": "^1.32.0",  
 "cors": "^2.8.5",  
 "dotenv": "^16.0.0",  
 "express": "^4.18.2",  
 "jsonwebtoken": "^9.0.0",  
 "multer": "^1.4.5-lts.1",  
 "multer-storage-cloudinary": "^4.0.0",  
 "sqlite3": "^5.1.6",  
 "pdfkit": "^0.13.0"  
 }  
}

### Backend .env.example

PORT=4000  
JWT\_SECRET=change\_this\_to\_a\_strong\_secret  
CLOUDINARY\_CLOUD\_NAME=your\_cloud\_name  
CLOUDINARY\_API\_KEY=your\_api\_key  
CLOUDINARY\_API\_SECRET=your\_api\_secret

### backend/db.js

const sqlite3 = require('sqlite3').verbose();  
const db = new sqlite3.Database('./oralvis.db');  
  
db.serialize(() => {  
 db.run(`  
 CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 email TEXT UNIQUE,  
 password TEXT,  
 role TEXT  
 )  
 `);  
  
 db.run(`  
 CREATE TABLE IF NOT EXISTS scans (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 patientName TEXT,  
 patientId TEXT,  
 scanType TEXT,  
 region TEXT,  
 imageUrl TEXT,  
 uploadDate TEXT  
 )  
 `);  
  
 // create sample users if not exists (passwords hashed in server start)  
});  
  
module.exports = db;

### backend/utils/cloudinary.js

const cloudinary = require('cloudinary').v2;  
const { CLOUDINARY\_API\_KEY, CLOUDINARY\_API\_SECRET, CLOUDINARY\_CLOUD\_NAME } = process.env;  
  
cloudinary.config({  
 cloud\_name: CLOUDINARY\_CLOUD\_NAME,  
 api\_key: CLOUDINARY\_API\_KEY,  
 api\_secret: CLOUDINARY\_API\_SECRET,  
});  
  
module.exports = cloudinary;

### backend/middleware/auth.js

const jwt = require('jsonwebtoken');  
const { JWT\_SECRET } = process.env;  
  
function authenticate(req, res, next) {  
 const auth = req.headers.authorization;  
 if (!auth) return res.status(401).json({ error: 'Missing token' });  
 const token = auth.split(' ')[1];  
 try {  
 const payload = jwt.verify(token, JWT\_SECRET);  
 req.user = payload;  
 next();  
 } catch (e) {  
 res.status(401).json({ error: 'Invalid token' });  
 }  
}  
  
function authorizeRole(role) {  
 return (req, res, next) => {  
 if (!req.user || req.user.role !== role) return res.status(403).json({ error: 'Forbidden' });  
 next();  
 };  
}  
  
module.exports = { authenticate, authorizeRole };

### backend/routes/auth.js

const express = require('express');  
const router = express.Router();  
const db = require('../db');  
const bcrypt = require('bcrypt');  
const jwt = require('jsonwebtoken');  
const { JWT\_SECRET = 'secret' } = process.env;  
  
// Register endpoint (optional utility)  
router.post('/register', async (req, res) => {  
 const { email, password, role } = req.body;  
 const hashed = await bcrypt.hash(password, 10);  
 db.run('INSERT INTO users (email, password, role) VALUES (?, ?, ?)', [email, hashed, role], function (err) {  
 if (err) return res.status(400).json({ error: err.message });  
 res.json({ id: this.lastID });  
 });  
});  
  
// Login  
router.post('/login', (req, res) => {  
 const { email, password } = req.body;  
 db.get('SELECT \* FROM users WHERE email = ?', [email], async (err, user) => {  
 if (err) return res.status(500).json({ error: err.message });  
 if (!user) return res.status(401).json({ error: 'Invalid credentials' });  
 const ok = await bcrypt.compare(password, user.password);  
 if (!ok) return res.status(401).json({ error: 'Invalid credentials' });  
 const token = jwt.sign({ id: user.id, email: user.email, role: user.role }, JWT\_SECRET, { expiresIn: '8h' });  
 res.json({ token, role: user.role });  
 });  
});  
  
module.exports = router;

### backend/routes/scans.js

const express = require('express');  
const router = express.Router();  
const multer = require('multer');  
const cloudinary = require('../utils/cloudinary');  
const { CloudinaryStorage } = require('multer-storage-cloudinary');  
const db = require('../db');  
const { authenticate, authorizeRole } = require('../middleware/auth');  
const PDFDocument = require('pdfkit');  
  
const storage = new CloudinaryStorage({  
 cloudinary: cloudinary,  
 params: {  
 folder: 'oralvis\_scans',  
 allowed\_formats: ['jpg', 'png', 'jpeg']  
 }  
});  
const upload = multer({ storage });  
  
// Upload (Technician only)  
router.post('/upload', authenticate, authorizeRole('technician'), upload.single('image'), (req, res) => {  
 const { patientName, patientId, scanType, region } = req.body;  
 const imageUrl = req.file.path || req.file.location || req.file?.path;  
 const uploadDate = new Date().toISOString();  
 db.run(  
 'INSERT INTO scans (patientName, patientId, scanType, region, imageUrl, uploadDate) VALUES (?, ?, ?, ?, ?, ?)',  
 [patientName, patientId, scanType, region, imageUrl, uploadDate],  
 function (err) {  
 if (err) return res.status(500).json({ error: err.message });  
 res.json({ id: this.lastID });  
 }  
 );  
});  
  
// List (Dentist only)  
router.get('/', authenticate, authorizeRole('dentist'), (req, res) => {  
 db.all('SELECT \* FROM scans ORDER BY uploadDate DESC', [], (err, rows) => {  
 if (err) return res.status(500).json({ error: err.message });  
 res.json(rows);  
 });  
});  
  
// Get single scan (Dentist) + PDF generation  
router.get('/:id', authenticate, authorizeRole('dentist'), (req, res) => {  
 const id = req.params.id;  
 db.get('SELECT \* FROM scans WHERE id = ?', [id], (err, row) => {  
 if (err) return res.status(500).json({ error: err.message });  
 if (!row) return res.status(404).json({ error: 'Not found' });  
 res.json(row);  
 });  
});  
  
// PDF endpoint  
router.get('/:id/pdf', authenticate, authorizeRole('dentist'), (req, res) => {  
 const id = req.params.id;  
 db.get('SELECT \* FROM scans WHERE id = ?', [id], async (err, row) => {  
 if (err) return res.status(500).json({ error: err.message });  
 if (!row) return res.status(404).json({ error: 'Not found' });  
  
 const doc = new PDFDocument();  
 res.setHeader('Content-type', 'application/pdf');  
 res.setHeader('Content-disposition', `attachment; filename=scan\_${id}.pdf`);  
  
 doc.pipe(res);  
 doc.fontSize(18).text('OralVis Scan Report', { align: 'center' });  
 doc.moveDown();  
 doc.fontSize(12).text(`Patient Name: ${row.patientName}`);  
 doc.text(`Patient ID: ${row.patientId}`);  
 doc.text(`Scan Type: ${row.scanType}`);  
 doc.text(`Region: ${row.region}`);  
 doc.text(`Upload Date: ${row.uploadDate}`);  
 doc.moveDown();  
  
 // Embed image (fetch from url)  
 try {  
 // PDFKit can take URL if install `image-downloader` or we can use request; keeping simple: use image from cloudinary via URL  
 doc.image(row.imageUrl, { fit: [450, 300], align: 'center' });  
 } catch (e) {  
 doc.text('Image could not be embedded.');  
 }  
  
 doc.end();  
 });  
});  
  
module.exports = router;

### backend/server.js

require('dotenv').config();  
const express = require('express');  
const cors = require('cors');  
const db = require('./db');  
const bcrypt = require('bcrypt');  
const authRoutes = require('./routes/auth');  
const scansRoutes = require('./routes/scans');  
  
const app = express();  
app.use(cors());  
app.use(express.json());  
  
app.use('/api/auth', authRoutes);  
app.use('/api/scans', scansRoutes);  
  
// create sample users if missing  
(async () => {  
 const techEmail = 'tech@example.com';  
 const dentistEmail = 'dentist@example.com';  
 db.get('SELECT \* FROM users WHERE email = ?', [techEmail], async (err, row) => {  
 if (!row) {  
 const hashed = await bcrypt.hash('password', 10);  
 db.run('INSERT INTO users (email, password, role) VALUES (?, ?, ?)', [techEmail, hashed, 'technician']);  
 }  
 });  
 db.get('SELECT \* FROM users WHERE email = ?', [dentistEmail], async (err, row) => {  
 if (!row) {  
 const hashed = await bcrypt.hash('password', 10);  
 db.run('INSERT INTO users (email, password, role) VALUES (?, ?, ?)', [dentistEmail, hashed, 'dentist']);  
 }  
 });  
})();  
  
const PORT = process.env.PORT || 4000;  
app.listen(PORT, () => console.log('Server running on', PORT));

## Frontend

### frontend/package.json

{  
 "name": "oralvis-frontend",  
 "version": "1.0.0",  
 "private": true,  
 "dependencies": {  
 "axios": "^1.4.0",  
 "react": "^18.2.0",  
 "react-dom": "^18.2.0",  
 "react-router-dom": "^6.14.1",  
 "jspdf": "^2.5.1"  
 },  
 "devDependencies": {  
 "vite": "^5.0.0"  
 },  
 "scripts": {  
 "dev": "vite",  
 "build": "vite build",  
 "preview": "vite preview"  
 }  
}

### frontend/.env.example

VITE\_API\_BASE=http://localhost:4000/api

### frontend/src/api.js

import axios from 'axios';  
  
const API\_BASE = import.meta.env.VITE\_API\_BASE || 'http://localhost:4000/api';  
  
const api = axios.create({ baseURL: API\_BASE });  
  
export function setAuthToken(token) {  
 if (token) api.defaults.headers.common['Authorization'] = `Bearer ${token}`;  
 else delete api.defaults.headers.common['Authorization'];  
}  
  
export default api;

### frontend/src/main.jsx

import React from 'react'  
import { createRoot } from 'react-dom/client'  
import { BrowserRouter, Routes, Route } from 'react-router-dom'  
import App from './App'  
import Login from './pages/Login'  
import TechnicianUpload from './pages/TechnicianUpload'  
import DentistViewer from './pages/DentistViewer'  
  
createRoot(document.getElementById('root')).render(  
 <React.StrictMode>  
 <BrowserRouter>  
 <Routes>  
 <Route path='/' element={<App/>}>  
 <Route index element={<Login/>} />  
 <Route path='upload' element={<TechnicianUpload/>} />  
 <Route path='viewer' element={<DentistViewer/>} />  
 </Route>  
 </Routes>  
 </BrowserRouter>  
 </React.StrictMode>  
)

### frontend/src/App.jsx

import React from 'react'  
import { Outlet, Link } from 'react-router-dom'  
import { setAuthToken } from './api'  
  
export default function App(){  
 // quick nav  
 return (  
 <div style={{ padding: 20 }}>  
 <nav>  
 <Link to='/'>Login</Link> | <Link to='/upload'>Technician Upload</Link> | <Link to='/viewer'>Dentist Viewer</Link>  
 </nav>  
 <hr/>  
 <Outlet />  
 </div>  
 )  
}

### frontend/src/pages/Login.jsx

import React, { useState } from 'react'  
import api, { setAuthToken } from '../api'  
  
export default function Login(){  
 const [email, setEmail] = useState('');  
 const [password, setPassword] = useState('');  
 const [role, setRole] = useState('');  
 const [msg, setMsg] = useState('');  
  
 const submit = async e => {  
 e.preventDefault();  
 try{  
 const res = await api.post('/auth/login', { email, password });  
 setAuthToken(res.data.token);  
 localStorage.setItem('token', res.data.token);  
 localStorage.setItem('role', res.data.role);  
 setMsg('Logged in as ' + res.data.role);  
 }catch(err){  
 setMsg(err?.response?.data?.error || 'Login failed');  
 }  
 }  
  
 return (  
 <form onSubmit={submit} style={{ maxWidth: 420 }}>  
 <h2>Login</h2>  
 <input value={email} onChange={e=>setEmail(e.target.value)} placeholder='Email' />  
 <br/>  
 <input type='password' value={password} onChange={e=>setPassword(e.target.value)} placeholder='Password' />  
 <br/>  
 <button type='submit'>Login</button>  
 <div>{msg}</div>  
 <p>Sample users: tech@example.com / password (technician), dentist@example.com / password (dentist)</p>  
 </form>  
 )  
}

### frontend/src/pages/TechnicianUpload.jsx

import React, { useState } from 'react'  
import api from '../api'  
  
export default function TechnicianUpload(){  
 const [form, setForm] = useState({ patientName: '', patientId: '', scanType: 'RGB', region: 'Frontal' });  
 const [file, setFile] = useState(null);  
 const [msg, setMsg] = useState('');  
  
 const submit = async e => {  
 e.preventDefault();  
 try{  
 const fd = new FormData();  
 fd.append('patientName', form.patientName);  
 fd.append('patientId', form.patientId);  
 fd.append('scanType', form.scanType);  
 fd.append('region', form.region);  
 fd.append('image', file);  
 const res = await api.post('/scans/upload', fd, { headers: { 'Content-Type': 'multipart/form-data' } });  
 setMsg('Uploaded, id: ' + res.data.id);  
 }catch(err){  
 setMsg(err?.response?.data?.error || 'Upload failed');  
 }  
 }  
  
 return (  
 <div>  
 <h2>Technician Upload</h2>  
 <form onSubmit={submit}>  
 <input value={form.patientName} onChange={e=>setForm({...form, patientName: e.target.value})} placeholder='Patient Name' />  
 <br/>  
 <input value={form.patientId} onChange={e=>setForm({...form, patientId: e.target.value})} placeholder='Patient ID' />  
 <br/>  
 <label>Scan Type</label>  
 <select value={form.scanType} onChange={e=>setForm({...form, scanType: e.target.value})}>  
 <option>RGB</option>  
 </select>  
 <br/>  
 <label>Region</label>  
 <select value={form.region} onChange={e=>setForm({...form, region: e.target.value})}>  
 <option>Frontal</option>  
 <option>Upper Arch</option>  
 <option>Lower Arch</option>  
 </select>  
 <br/>  
 <input type='file' accept='image/\*' onChange={e=>setFile(e.target.files[0])} />  
 <br/>  
 <button type='submit'>Upload</button>  
 </form>  
 <div>{msg}</div>  
 </div>  
 )  
}

### frontend/src/pages/DentistViewer.jsx

import React, { useEffect, useState } from 'react'  
import api from '../api'  
  
export default function DentistViewer(){  
 const [scans, setScans] = useState([]);  
  
 useEffect(()=>{  
 async function load(){  
 try{  
 const res = await api.get('/scans');  
 setScans(res.data);  
 }catch(err){  
 console.error(err);  
 }  
 }  
 load();  
 },[]);  
  
 return (  
 <div>  
 <h2>Scans</h2>  
 <div style={{ display: 'grid', gridTemplateColumns: 'repeat(3, 1fr)', gap: 12 }}>  
 {scans.map(s => (  
 <div key={s.id} style={{ border: '1px solid #ddd', padding: 8 }}>  
 <h4>{s.patientName}</h4>  
 <div>ID: {s.patientId}</div>  
 <div>Type: {s.scanType}</div>  
 <div>Region: {s.region}</div>  
 <div>Date: {new Date(s.uploadDate).toLocaleString()}</div>  
 <img src={s.imageUrl} alt='' style={{ width: '100%', maxHeight: 150, objectFit: 'cover' }} />  
 <a href={s.imageUrl} target='\_blank'>View full image</a>  
 <br/>  
 <a href={`/api/scans/${s.id}/pdf`} target='\_blank'>Download PDF</a>  
 </div>  
 ))}  
 </div>  
 </div>  
 )  
}

## .gitignore

node\_modules/  
.env  
/frontend/node\_modules/  
/backend/node\_modules/  
oralvis.db  
dist/

## README.md (root)

# OralVis Healthcare  
  
Full-stack sample app for Technician (upload scans) and Dentist (view & download PDFs).  
  
## Tech stack  
- Backend: Node.js, Express, SQLite  
- Frontend: React (Vite)  
- Cloud storage: Cloudinary (recommended)  
- PDF: PDFKit  
  
## Local setup  
  
### Backend  
1. `cd backend`  
2. `cp .env.example .env` and fill values  
3. `npm install`  
4. `npm run dev` (or `npm start`)  
  
### Frontend  
1. `cd frontend`  
2. `cp .env.example .env` (change API base if backend deployed)  
3. `npm install`  
4. `npm run dev`  
  
## Deployment  
- Backend: Render / Heroku / Railway. Set env vars in the service dashboard.  
- Frontend: Vercel or Netlify. Set `VITE\_API\_BASE` to the deployed backend URL.  
  
## Submission checklist  
- Push repository to GitHub.  
- Add screenshots to README.  
- Add hosted demo link.

### How to upload to GitHub (quick steps)

1. git init
2. git add .
3. git commit -m "Initial commit: OralVis Healthcare"
4. Create a repo on GitHub web, then follow the remote add + push commands shown there, e.g.:
   * git remote add origin git@github.com:yourusername/oralvis-healthcare.git
   * git branch -M main
   * git push -u origin main

## Notes & Caveats

* The backend creates two sample users on start: tech@example.com (password password, role technician) and dentist@example.com (password password, role dentist).
* For Cloudinary: set credentials in .env. The multer-storage-cloudinary plugin is used for easy upload.
* PDFKit embedding of remote images may fail for some remote URLs due to required fetching; if that happens, consider downloading the file server-side (e.g., with node-fetch) before piping to PDFKit.

If you want, I can now: - produce a ready .zip with the files, OR - push this structure into a GitHub repo for you (I’ll provide step-by-step commands), OR - generate missing helper files (e.g., Vite index.html) — pick one and I’ll continue.