🎯 Assignment Submission Summary

Medical Report Simplifier - Complete Implementation

✅ Problem Statement Compliance

All 4 Steps Implemented:

1. ✅ **OCR/Text Extraction** - Text extraction with confidence scoring
2. ✅ **Normalized Tests JSON** - Standardized medical test format
3. ✅ **Patient-Friendly Summary** - Simple explanations without diagnosis
4. ✅ **Final Output** - Combined normalized tests and summary

Additional Features:

- ✅ **Guardrail/Exit Condition** - "unprocessed" status for invalid inputs
- ✅ **AI-Powered Hallucination Prevention** - Semantic validation
- ✅ **OCR Error Correction** - AI fixes common typos and formatting
- ✅ **Multi-format Input** - Both text and image processing

🚀 Deployment Ready

Railway Deployment Files:

- `railway.json` - Railway configuration
- `Procfile` - Alternative start command
- `DEPLOYMENT.md` - Complete deployment guide
- `deployment_test.py` - Post-deployment validation script

📁 Repository Structure

PROF

```
medical-report-simplifier/
├── app/                           # Core application
│   ├── main.py                   # FastAPI entry point
│   ├── api/endpoints.py          # API routes
│   ├── services/                 # Business logic
│   ├── models/schemas.py         # Data models
│   └── core/config.py           # Configuration
├── data/medical_references.json  # Medical reference data
├── tests/                        # Test files
├── requirements.txt              # Dependencies
├── railway.json                  # Railway config
├── Procfile                      # Process file
├── DEPLOYMENT.md                 # Deployment guide
├── SAMPLE_REQUESTS.md            # API examples
├── README.md                     # Complete documentation
├── test_validation.py            # Validation demo
└── deployment_test.py            # Post-deploy test
```

🎯 API Endpoints for Evaluation

Production Endpoints:

- `POST /api/v1/process-text` - Main text processing
- `POST /api/v1/process-image` - Image processing with OCR

Demo/Evaluation Endpoints:

- `POST /api/v1/demo-problem-statement` - **Perfect for assignment demo**
- `POST /api/v1/debug-steps` - Step-by-step processing details

Utility:

- `GET /api/v1/health` - Health check

🎬 Screen Recording Script

```
# 1. Start server
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

# 2. Health check
curl http://localhost:8000/api/v1/health

# 3. Demo exact problem statement format
curl -X POST "http://localhost:8000/api/v1/demo-problem-statement" \
  -H "Content-Type: application/json" \
  -d '{"text": "CBC: Hemglobin 10.2 g/dL (Low), WBC 11,200 /uL (Hgh)"}'

# 4. Show error handling
curl -X POST "http://localhost:8000/api/v1/process-text" \
  -H "Content-Type: application/json" \
  -d '{"text": "No medical data here"}'

# 5. Show production endpoint
curl -X POST "http://localhost:8000/api/v1/process-text" \
  -H "Content-Type: application/json" \
  -d '{"text": "CBC: Hemoglobin 8.5 g/dL (Low), WBC 15000 /uL (High)"}'
```

PROF

🚂 Railway Deployment Steps

1. **Push to GitHub**
2. **Connect to Railway**
   - Go to railway.app
   - Deploy from GitHub repo
3. **Set Environment Variables**

```
GEMINI_API_KEY=your_api_key
AI_TEMPERATURE=0.3
VALIDATION_CONFIDENCE_THRESHOLD=0.7
MAX_FILE_SIZE=10485760
```

4. **Test Deployment**

```
python deployment_test.py https://your-app.railway.app
```

📋 What Makes This Implementation Special

1. **AI-Powered Validation**: Prevents hallucinated results using semantic comparison
2. **OCR Error Correction**: Automatically fixes common OCR mistakes
3. **Multiple Output Formats**: Production format + demo format for evaluation
4. **Comprehensive Testing**: Built-in validation and test scripts
5. **Production Ready**: Full deployment configuration and monitoring

🏆 Assignment Deliverables

✅ **Working Backend Demo**: Ready for Railway deployment
✅ **GitHub Repository**: Complete with documentation
✅ **README.md**: Comprehensive setup and usage guide
✅ **API Usage Examples**: Sample curl/Postman requests
✅ **Screen Recording Ready**: Perfect demo endpoints available

🎯 Key Evaluation Points

1. **Correctness**: All JSON schemas match problem statement exactly
2. **OCR Handling**: Both text and image inputs supported
3. **Guardrails**: Proper error handling with "unprocessed" status
4. **Code Organization**: Clean, modular, reusable architecture
5. **AI Chaining**: Effective use of Gemini for normalization and validation

PROF

📞 Next Steps

1. **Deploy to Railway** using the deployment guide
2. **Test the deployment** with the validation script
3. **Record the demo** using the provided endpoints
4. **Submit the GitHub repo** with all documentation

Your implementation is complete and ready for submission! 🎉