**📄 Telemedicine Backend Development Summary**

**✅ Project Overview**

You created a **microservices backend** with:

* **auth-service** (authentication & token issuance)
* **user-service** (profile data)
* **MongoDB** (shared database)

**✅ Services & Ports**

|  |  |  |
| --- | --- | --- |
| Service | Port | Purpose |
| auth-service | 5001 | Register/Login users |
| user-service | 5002 | Store/Retrieve user profiles |
| MongoDB | 27017 | Database storage |

**✅ Docker Compose File**

**docker-compose.yml**

version: '3.8'

services:

auth:

build: ./auth

ports:

- "5001:5001"

env\_file:

- ./auth/.env

volumes:

- ./auth:/app

depends\_on:

- mongo

user:

build: ./user-service

ports:

- "5002:5002"

env\_file:

- ./user-service/.env

volumes:

- ./user-service:/app

depends\_on:

- mongo

mongo:

image: mongo

container\_name: mongodb

ports:

- "27017:27017"

volumes:

- mongo\_data:/data/db

volumes:

mongo\_data:

**✅ .env Files**

**auth/.env**

PORT=5001

MONGO\_URI=mongodb://mongo:27017/authdb

JWT\_SECRET=your\_super\_secret\_key

**user-service/.env**

PORT=5002

MONGO\_URI=mongodb://mongo:27017/userdb

JWT\_SECRET=your\_super\_secret\_key

**✅ Mongoose Models**

**auth-service/src/models/User.js**

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

name: { type: String, required: true },

email: { type: String, required: true },

password: { type: String, required: true },

role: { type: String, required: true }

});

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

**user-service/src/models/User.js**

const mongoose = require('mongoose');

const UserData = new mongoose.Schema({

firstName: { type: String, required: true },

lastName: { type: String, required: true },

gender: { type: String, required: true },

dob: { type: Date, required: true },

email: { type: String, required: true },

address: { type: String, required: true },

role: { type: String, enum: ['user', 'doctor', 'admin'], default: 'user' }

});

module.exports = mongoose.model('userdata', UserData);

**✅ JWT Authentication**

**auth-service/src/routes/auth.js**

const express = require('express');

const router = express.Router();

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const User = require('../models/User');

// Register

router.post('/register', async (req, res) => {

const { name, email, password, role } = req.body;

const existingUser = await User.findOne({ email });

if (existingUser) return res.status(400).json({ message: 'User already exists' });

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

const newUser = new User({ name, email, password: hashedPassword, role });

await newUser.save();

res.status(201).json({ message: 'User registered successfully' });

});

// Login

router.post('/login', async (req, res) => {

const { email, password } = req.body;

const user = await User.findOne({ email });

if (!user) return res.status(400).json({ message: 'Invalid credentials' });

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });

const token = jwt.sign(

{ id: user.\_id, role: user.role },

process.env.JWT\_SECRET,

{ expiresIn: '1d' }

);

res.status(200).json({

token,

user: { id: user.\_id, name: user.name, role: user.role }

});

});

module.exports = router;

**user-service/src/middleware/authMiddleware.js**

const jwt = require('jsonwebtoken');

const protect = (req, res, next) => {

const authHeader = req.headers.authorization;

if (!authHeader || !authHeader.startsWith('Bearer ')) {

return res.status(401).json({ message: 'Unauthorized: No token' });

}

const token = authHeader.split(' ')[1];

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

req.user = decoded;

next();

} catch (err) {

return res.status(403).json({ message: 'Invalid token' });

}

};

module.exports = protect;

**✅ How to Get a Token**

**Login API:**

POST http://localhost:5001/api/auth/login

**Request Body:**

{

"email": "your\_email@example.com",

"password": "your\_password"

}

**Response:**

{

"token": "your\_jwt\_token",

"user": { ... }

}

**Use this token in Authorization header:**

Authorization: Bearer your\_jwt\_token

✅ **Register API:**

bash

CopyEdit

POST http://localhost:5001/api/auth/register

✅ **Request Body example:**

json

CopyEdit

{

"name": "paras",

"email": "parassa@example.com",

"password": "strongpassword",

"role": "user"

}

This will create a new user in your auth-service.

**✅ Testing User Service APIs**

**Get All Users (public)**

GET http://localhost:5002/api/users

**Get User By ID (protected)**

GET http://localhost:5002/api/users/<id>

**Headers:**

Authorization: Bearer your\_jwt\_token

**Create User Profile (protected)**

POST http://localhost:5002/api/users

**Headers:**

Authorization: Bearer your\_jwt\_token

Content-Type: application/json

**Body:**

{

"firstName": "Paras",

"lastName": "Sachdeva",

"gender": "Male",

"dob": "1995-06-15",

"email": "parassa@example.com",

"address": "India",

"role": "user"

}

**✅ Useful Docker Commands**

* **Build all services:**
* docker-compose build
* **Start all services:**
* docker-compose up
* **Stop and remove all services:**
* docker-compose down -v
* **View running containers:**
* docker ps

✅ **Tip:**  
Keep this document as your quick reference.  
If you want, I can help you extend it with more services or advanced features.

If you’d like, I can export this as Markdown or text file. Just let me know!