OTP VERIFICATION

Server.js

// server.js

const express = require('express');

const mongoose = require('mongoose');

require('dotenv').config();

const app = express();

app.use(express.json());

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then(() => console.log('MongoDB connected'))

.catch((err) => console.log(err));

// Import Routes

const otpRoutes = require('./routes/otpRoutes');

app.use('/api/otp', otpRoutes);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

.env

MONGO\_URI=mongodb://localhost:27017/otp-verification

PORT=5000

EMAIL\_USER=example@gmail.com

EMAIL\_PASS=app password

models/Otp.js

// models/Otp.js

const mongoose = require('mongoose');

const otpSchema = new mongoose.Schema({

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

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

createdAt: { type: Date, default: Date.now, index: { expires: 300 } } // expires in 5 minutes

});

module.exports = mongoose.model('Otp', otpSchema);

Routes /otpRoutes.js

// routes/otpRoutes.js

const express = require('express');

const router = express.Router();

const otpController = require('../controllers/otpController');

router.post('/send', otpController.sendOtp);

router.post('/verify', otpController.verifyOtp);

module.exports = router;

Controllers / otpController.js

// controllers/otpController.js

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

const crypto = require('crypto');

const nodemailer = require('nodemailer');

// Function to generate OTP

const generateOtp = () => crypto.randomInt(100000, 999999).toString();

// Nodemailer configuration

const transporter = nodemailer.createTransport({

service: 'gmail', // Use your email provider

auth: {

user: process.env.EMAIL\_USER,

pass: process.env.EMAIL\_PASS,

},

});

exports.sendOtp = async (req, res) => {

const { email } = req.body;

const otp = generateOtp();

try {

await Otp.deleteMany({ email }); // Remove old OTP if exists

const otpRecord = new Otp({ email, otp });

await otpRecord.save();

// Send OTP via email

await transporter.sendMail({

from: process.env.EMAIL\_USER,

to: email,

subject: 'Your OTP Code',

text: `Your OTP code is ${otp}`,

});

res.status(200).json({ message: 'OTP sent successfully' });

} catch (error) {

res.status(500).json({ message: 'Error sending OTP', error });

}

};

exports.verifyOtp = async (req, res) => {

const { email, otp } = req.body;

try {

const otpRecord = await Otp.findOne({ email, otp });

if (!otpRecord) {

return res.status(400).json({ message: 'Invalid OTP or OTP expired' });

}

await Otp.deleteOne({ email, otp }); // Remove OTP after verification

res.status(200).json({ message: 'OTP verified successfully' });

} catch (error) {

res.status(500).json({ message: 'Error verifying OTP', error });

}

};