|  |  |
| --- | --- |
| **App.js**  const middlewares = require('./middlewares/app.middleware');  const errorHandler = require('../src/utils/errorHandler.util')  **// Loads .env file contents for local development.**  if (!process.env.APP\_ENV) require('dotenv').config();  **// db connection.**  const db = require('./configs/db.config');  db.initialize\_connection();  const app = require('express')();  **// Middlewares**  middlewares(app);  **// Routes**  app.use(process.env.API\_ENDPOINT + '/', require('./routes'));  **//Catch 404 and forward to error handler**  app.use('\*',(req, res, next) => {  const error = new Error(`Can't find ${req.originalUrl} on the server!`, 404);  error.statusCode = 404;  next(error);  app.use(errorHandler);  module.exports = app; | **App.middleware**  const express = require('express');  const helmet = require('helmet');  const cors = require('cors');  const rateLimiter = require('express-rate-limit');  const xss = require('xss-clean');  const mongoSanitize = require('express-mongo-sanitize');  var logger = require('morgan');  var path = require('path');  var cookieParser = require('cookie-parser');  module.exports = (app) => {  app.use(cors()); app.use(helmet());  app.use(helmet.crossOriginResourcePolicy({ policy: "cross-origin" }));  app.use(express.json({ limit: '20mb' }));  app.use(express.urlencoded({ extended: false }));  app.use(rateLimiter({  windowMs: 15 \* 60 \* 1000,max: 60, }));  app.use(xss());  app.use(mongoSanitize());  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));}; |

|  |  |
| --- | --- |
| **ErrorHandle.js**  const express = require('express');  const CustomError = require('../utils/customErrorHandler');  const response = require('../utils/responseHandlers.utils');  const devErrors = (res, error) => {  const person = {  status: error.statusCode,  message: error.message,  stackTrace: error.stack,  };  response.middleware\_Error  \_message(person,error,res)}  module.exports = (error, req, res, next) => {  error.statusCode = error.statusCode || 500;  error.status = error.status || 'error';  if(process.env.NODE\_ENV === 'development'){  devErrors(res, error);  } else if(process.env.NODE\_ENV === 'production'){  if(error.name === 'CastError') error = castErrorHandler(error);  if(error.code === 11000) error = duplicateKeyErrorHandler(error);  if(error.name === 'ValidationError') error = validationErrorHandler(error);  prodErrors(res, error); }}  **Router.js**  const router = require('express').Router();  router.get('/health', (\_, res) => {  return res.status(200).send('As Strong as an Ox!');});  router.use('/', require('./auth.route'));  router.use('/users', require('./user.route'));  module.exports = router;  **auth.route.js**  const router = require('express').Router();  const handler = require('../controllers/auth.controller')  const { createUserValidationRules} = require('../validations/express-validator');  router.post('/login',createUserValidationRules(), handler.login);  router.post('/register',createUserValidationRules(), handler.register);  module.exports = router; | **Service file.js**  const { v4: uuidv4 } = require('uuid');  const { validationResult } = require('express-validator');  const USER = require('../models/user.model');  const generate\_token = require('../helpers/generateToken.helper');  const { createUserValidationRules } = require('../validations/express-validator');  const response = require('../utils/responseHandlers.utils');  function register(req, res,next) {  try{  const errors = validationResult(req);  if (!errors.isEmpty()) {  return response.validation\_error\_message(errors.array(), res);  // return res.status(400).json({ errors: errors.array() });}  const payload = req.bodys;  const userData = {  firstName: payload.firstName,  lastName: payload.lastName,  password: payload.password,  gender: payload.gender,  email: payload.email,  secretOrKey: uuidv4(),  status: 'ACTIVE',};  const user = new USER(userData);  user  .save()  .then((\_doc) => {  const responseData = {  user: {  firstName: \_doc.firstName,  lastName: \_doc.lastName,  gender: \_doc.gender,  email: \_doc.email,  id: \_doc.\_id,},  token: generate\_token({ secretOrKey: \_doc.secretOrKey }), };  res.json(responseData); })  .catch((error) => {  res.status(500).json({ error }); });  }catch(error){  next(error);}}  module.exports = {  register}; |

|  |  |
| --- | --- |
| const { check } = require('express-validator');  const createUserValidationRules = () => {return [ check('username') .isLength({ min: 3 }).withMessage('Username must be at least 3 characters long'),  module.exports = {createUserValidationRules,}; | check('email')  .isEmail().withMessage('Please provide a valid email address'),  check('password') .isLength({ min: 6 }).withMessage('Password must be at least 6 characters long')];}; |