PECHKUR OLEKSANDR

NODE.JS / NEST.JS / AWS LAMBDA DEVELOPER

CONTACT

+380673417791

pechkursasha@gmail.com

Kyiv

Github

SKILLS

Languages: TypeScript, JavaScript (ES6+)
Backend: Node.js, Express.js, Nest.js
Database: MongoDB (Mongoose)
Auth & Security: JWT, RBAC, bcrypt
Cloud & Serverless: AWS Lambda, SES, S3, SQS
DevOps & Tools: dotenv, Git, GitHub, GitLab
Automation: node-cron, Railway
Testing & CI/CD: Jest, Docker, GitHub Actions

LANGUAGES

English Intermediate
Ukrainian (Fluent)

EDUCATION

Master's Degree in Management (Agile Software Development focus) [KROK, Kyiv | 2023–2025

PROFILE

Completed a 4-month backend mentorship under Nazar Zastavnyi (Tech Lead at Dashedvs).
Gained practical experience in clean code, backend architecture, and AWS deployment.

EXPERIENCE

Weather-bot(Backend Developer – Personal Project) 2025
Built and deployed a Node.js bot that provides users with real-time weather updates based on their geolocation.
Implemented daily weather notifications using node-cron, with support for custom scheduling and error handling.
Integrated OpenWeatherMap API and automated configuration via environment variables for secure deployment.
Used Telegraph for user interaction, Axios for API requests, and structured logs for monitoring.
Stack: Node.js, Axios, Open Weather Map API, node-cron, dotenv, Telegraph API

AWS-Contact-Form-Handler Backend Developer – Personal Project) 2025
(Developed a serverless function on AWS Lambda to handle contact form submissions in real-time.
Automated two-way email notifications: one to the admin (inbox alert) and one to the user (confirmation receipt).
Used AWS SES for transactional emails and S3 for backup logging of form submissions.
Secured access with IAM roles and environment-based config management (dotenv).
Manual deployment through AWS Console with permission-scoped IAM policies.
Stack: Node.js, AWS Lambda, AWS SES, AWS S3, dotenv, IAM

User Management System (Backend Developer – Freelance Project) 2025
Built a scalable user management system using NestJS and TypeScript, following clean architecture principles.
Implemented JWT-based authentication, role-based access control (RBAC), and secure password hashing with bcrypt.
Used class-validator and class-transformer to ensure strict data validation and transformation.
Structured project for future scalability with clear separation of modules, services, and guards.
Stack: NestJS, TypeScript, MongoDB (Mongoose), JWT, bcrypt, class-validator, dotenv, ESLint, Prettier