

MAHMOUD ZAKY FETOH

Phone: +201022381474

Email: zaky.fetoh@gmail.com

github.com/zaky-fetoh

linkedin.com/in/mahmoud-zaky-fetoh

Objective

I am a junior MERN stack engineer passionate about solving a real-world problem, working and collaborate with a great Engineers that I can learn from. I have strong knowledge of algorithms and writing efficient and high performing code.

Education

2020-Present	Faculty of Computers and Information, Menoufia university , Egypt. M.Sc., doing computer vision research, Expected graduation 2023
2015-2019	Faculty of Computers and Information, Menoufia university , Egypt. B.Sc. Honors, I graduated with the Top, 1st, Grade, Excellent with honor GPA 3.5.

Work Experience

2023-Present	Data Scientist Susoft , Norway Building microservice application for Training and deploying machine and deep learning models for Sales forecasting using. Performing customer segmentation to direct marketing campaigns. Technologies: Model training: PyTorchForecasting, Pytorch, pandas, Prophet, NeuralProphet. Model monitoring: Weight and biases. Model Serving: torchScript, Model Registry: Minio. Running Environment: Docker compose Asynchronous communication for issuing train request is done using RabbitMQ Gateway and load balancing services done using NodeJS.
--------------	---

Frameworks & Technical Skills

Languages:	Python, NodeJS, GoLang (learning)
Programming Paradigms:	Object-oriented programming (OOP) and Functional Programming (FP).
HTTP Servers Frameworks:	ExpressJS, Flask.
Front-End Frameworks:	ReactJS, Angular (learning).
Version Control Tools:	Git, Github, Bitbucket.
Client API:	REST, GraphQL, gRPC, SocketIO.
Databases:	MONGODB, MySQL, PostgreSQL, MariaDB.
Data Science:	PyTorch, Pandas, statsmodels, Prophet, Numpy, OpenCV, Plotly.
Security Utilities:	Joi, Jsonwebtoken (JWT).
DevOps:	CCNA, MCSA, Docker-Compose, Kubernetes (learning).
Message Broker:	RabbitMQ, Kafka (learning).
Caching:	Redis, MinIO.
Testing:	Jest.

Project Experience

- 2023 **Personal Blog Site**, [\[Link\]](#).
This is a MERN stack project that manage personal blog and store it in MONGODB database. It performs caching for query using redis. It's frontend is build using ReactJS and it state is managed using Redux-toolkit.
- Built with:* NodeJS, ExpressJS, ReactJS, Crypto, Redis, MongoDB, Docker-Compose, Redux-toolkit, formik, yup.
- 2022 **Clinic Management System**, [\[Link\]](#).

A backend System for managing clinics, it's department, medical stuff, and patients. This System provides a RESTfull, and GraphQL for clients. Complex mongodb aggregation pipeline is implemented for extracting high level info.

- *Built with:* NodeJS, Express, Mongoose, graphql, Joi.

2022 **Image Manager**, [\[Link\]](#).

Microservices application for image storage, stored in plain form or encrypted form, with it's meta data and automatic tagging using Deep learning techniques.

It consists of three microservices as follow:

- **Image-Classification micro-service:** This micro-service is responsible for classifying images using ResNet18 other services can communicate with it using synchronous communication using by requesting it RESTfull API or through asynchronous communication using RabbitMQ.
 - *Built with:* Python, Pytorch, Flask, pika.
- **Image-Storage micro-service:** This micro-service is responsible for storing the images the users in either plain or encrypted, using AES, form other microservices can communicate with it using it's RESTfull API.
 - *Built with:* NodeJS, Express, Multer, crypto.
- **Image-manager micro-service:** This micro-service is responsible for 1) storing the meta-data, such as Image's owner and the image Tags, automatically added by Image Classification micro-service, and the total number of views, 2) manage users, users credentials and authentication 3) communicate with Image-Classification microservice using RabbitMQ and With Image-Storage through it's RESTfull API.
 - *Built with:* NodeJS, Express, Mongoose, JWT, bcrypt, amqplib, Joi.
- **Route-Advertisement micro-service:** To avoid manually adjust API Gateway for adding public route route-advertiment (RA) manages all routes for the entire micro-service application. Any service wants to advertise a specific public route it can HTTP POST this route to RA through it's RESTfull API. RA assumes order of routes does not matter. RA allows dynamically adding routes to the API Gateway.
 - *Built with:* NodeJS, Express, Mongoose, axios, JWT.
- **API Gateway micro-service:** API Gateway (GW) micro-service periodically request new added routes to route-advertisement. Every GW is responsible of keep tracking it's state and request missed routes.
 - *Built with:* NodeJS, Express, Mongoose, axios, cron.

2022 **Notification Server**, [\[Link\]](#).

Periodically send a scheduled Notification using scheduled using CronJob.