Darshan Sharma

+91-7009746321 thedarshansharma@gmail.com https://linkedin.com/in/darshansharmain

Education

Panjab University
Bachelors in Engineering (Computer Science) 2014 - 2018

Skills

Web: HTML5, CSS3
Frameworks: **React.Js**, **Node.Js**, Express, Nest.Js, Flutter
Tech Stack: **MERN**, LAMP, PERN
DevOps: GitHub Actions, Workflow, CI/CD Pipeline
Cloud: **AWS-Lambda-Functions**, EC2, ECS, EKS, S3, **Docker**, Kubernetes
Web API: REST API, **GraphQL**, SOAP
Database: PostgreSQL, MongoDB, Redis
Load Balancer: RabbitMQ, AmazonMQ
Prog. Languages: Java, Python, C++, JavaScript, TypeScript
Operating System: AGL, ArchLinux
Artificial Intelligence: **GenAI**, LLM-finetuning, CNN, NLP, AWS-Bedrock

Work Experience (5+ Years)

MonkAI Technologies

Position: Senior Software Engineer **May 2020 - Present**

- Improved React video player load time in the app feed by pre-loading just 25% of the content, fetching the rest only after 90% viewership of the initial segment, guided by user data analysis
  Techstack: TypeScript, React, Node, AWS-Lambda, S3, MongoDB

- Reduced DB expenses by 10% by integrating Redis as a caching layer, resulting in a single definitive database call rather than 6-7 heavy Mongodb calls.
  Techstack: React, Nest, AWS-RDS, GraphQL, TypeScript

- Developed 9 Flutter mobile and web apps featuring in-app purchases, animations, push alerts, maps, auth. Released them on Play/App Store
  Techstack: Flutter, Nest, Firebase, Lambda-Function, GraphQL

Paxcom

Position: Software Engineer **May 2019 - Oct, 2019**

- Decreased the load on the node server backend by 70% as it was being used for heavy CPU computations like image processing, and CSV to HTML conversion using a RabbitMQ as a load balancer
  Techstack: React, Node, EC2, TypeScript, AmazonMQ, Lambda-Functions

- Achieved a 30% decrease in server expenses by transitioning from a monolithic to microservices, using RabbitMQ for load distribution, coupled with the integration of automated scaling and efficient traffic management.

Block8

Position: Software Engineer **May 2018 - May, 2019**

- Improved React-Node app's speed by tweaking database queries, majorly enhancing its operational efficiency. Reduced LCP from 6s to 3s using CDN to serve static assets.
  Techstack: React, Node, EC2, TypeScript, MongoDB

Projects

Project name: Block8 website
Tech Stack: React, Node, AWS-EC2, TypeScript, MongoDB
Project Link: https://www.mystake.io
At Block8 we were using React for the front end and Node for the back end. The database queries, which were written inside Lambda functions, were not optimized, performing the cross-operation first and then the selection operation. I tweaked it a bit to perform the selection operation first, followed by joins, and then the final selection. In this way, we were able to reduce some latency.

Project name: Paxcom backend part
Tech Stack: Node, RabbitMQ, Microservices, TypeScript, EC2, ECS
Monthly Active Users: B2B & B2C: 30,000 requests per day
Project Link: https://www.paxcom.ai
Reduced the memory usage by 70% of a CPU-intensive task of Excel to HTML conversion in Node.js by architecting a dedicated microservice separately for it. At Paxcom, my task was to alleviate the backend server load. I transitioned from a monolithic structure to microservices, addressing the issue of a single node being swamped by both database operations and CPU-heavy tasks like B2B Excel to HTML and PDF generation. By adopting a microservice approach and leveraging AmazonMQ, tasks were efficiently distributed based on request type, optimizing server performance

Project name: apti
Teck Stack used: Flutter, Nest.Js, React, Firebase Auth, MongoDB, Redis, GraphQL, TypeScript
Monthly Active Users: 50,000 incoming requests per day at peak time during Covid-19
Project Link: Released on App/Play Store
At the time of COVID-19, the company wanted to launch an ed-tech-based product, in which Aptitude-based questions within a set timer running to test the cognitive, aptitude, psychomotor, numerical reasoning, ability to adapt, soft skills, emotional intelligence, critical thinking, and attentiveness these skills were tested. A lot of exercises were created by myself at the start. How much time candidate take to tap for user telemetry and further optimizations were also performed.

Project name: keytome (UK-Client)
Tech Stack used: React, Nest.Js, GraphQL, TypeScript, Flutter, AWS-S3, CloudFront
Number of Users: 20,000 monthly active users
Project Link: Released on App/Play Store
To reduce the initial loading time. Videos were stored in S3 and their simultaneous display on the home feed, even with CloudFront caching, was bandwidth and time-intensive. To optimize, we split videos into 4 or 8 segments based on duration. Initially, only the first segment is sent to users; subsequent parts are delivered only if 90% of the current segment is viewed. This approach, informed by MongoDB analytics, efficiently manages large video files on AWS S3 and CloudFront.

Project name: BookMate
Tech Stack used: Claude Generative AI, AWS
Project Link: https://partyrock.aws/bookmate
A study companion, designed to assist students during the crucial final moments before their exam. If they have just one day left to prepare and haven't started preparation for examination yet.