Vikalp Pandey

+91 8882323639 | vikalp.pandey2004@gmail.com | **LinkedIn** | **GitHub** | **Website**

EDUCATION

Bharati Vidyapeeth College of Engineering | B.Tech, ECE	8.05 CGPA	*July 2025*
SCSD Sarvodaya Vidyalaya | Class 12	77%	*July 2021*
SD Saraswati Bal Mandir | Class 10	92.17%	*April 2019*

PROFESSIONAL EXPERIENCE

SequelString AI | Software Engineer | Remote *Nov 2025 to Present*

- Developed a **Metro reconciliation application** using **React, FastAPI, and OpenPyXL,** consolidating repeated Excel-based tasks into a workflow for **uploads, validation, calculations, and result downloads.**
- Automated **buyer payment-reminder emails** for **pending and short payments** flagged by **negative closing balances,** validated receipts against **bank statements,** and calculated **cumulative amounts settled to each corresponding Metro.**
- Automated **GSTR-2A invoice reconciliation** by normalizing **invoice numbers** and combining **tax values** to match records, then generated Excel reports separating **matched invoices from unmatched entries.**
- Built **bank-statement validation and balance calculation** to check dates, narration, debit and credit values, **duplicate entries,** and **running balances** before calculating the **closing balance.**
- Implemented **REST APIs and reconciliation safeguards** for **multi-file uploads, DRAFT/CALCULATED/REJECTED status tracking,** and **read-only storage** of source files.
- Created **responsive React interfaces** for run creation, file uploads, **validation results,** and **file-level calculations,** with **actionable error messages** to help users correct invalid data.
- Developed a **refund-management portal** for **DMRC and the SequelString Finance team** to track buyer payments, **cancellation status, refund initiation,** and **settlement progress** through one centralized workflow.
- Automated **buyer refund notifications** based on **cancellation status** and tracked **pending, processed, and received refunds,** improving follow-up and **financial visibility.**

PROJECTS

StoreX | Cloud File Storage & Sharing Platform Video Demo | GitHub | Link

Tech: *TypeScript, Hono, MongoDB, Redis, AWS S3/SQS/Lambda, OpenFGA, Stripe, SST*

- Engineered **direct-to-S3 uploads** for **files larger than 5 GB** using **presigned multipart transfers** that send files directly from the browser to S3; tracked upload parts and managed completion or cancellation **without routing file data through the API**.
- Designed **hierarchical file-management APIs** with MongoDB models for nested folders, paginated listings, search, and **trash/restore**; automated permanent deletion of files kept in trash for **over 30 days** using a **daily cron job**.
- Implemented **fine-grained access control** for sharing nested files and folders with **OpenFGA relationship-based permissions** and **inherited access**, connecting **Google/GitHub OAuth** identities to authorization checks for shared resources.
- Designed **CDN-backed file delivery** with layered caching to serve cached S3 content from the CDN, **reduce repeated origin fetches**, and use **Redis and TanStack Query** to limit redundant database reads and client API requests.
- Connected subscription billing to **storage entitlements** through **Stripe plans and verified webhooks**, keeping account **storage quotas** aligned with billing events and subscription status.
- Decoupled **signup email delivery** with **Amazon SQS**, processing emails **asynchronously** from the signup workflow.
- Codified the AWS stack using **SST infrastructure as code**, defining storage, queues, compute, and file delivery resources in **sst.config.ts** for repeatable provisioning from a TypeScript configuration.

StreamForge | Adaptive Video Transcoding & HLS Streaming Platform Video Demo | GitHub | Link

Tech: *TypeScript, React, Hono, MongoDB, AWS S3/SQS/Lambda/ECS/CloudFront, FFmpeg, Video.js, SST*

- Engineered **resumable direct-to-S3 video uploads** using presigned URLs and **10 MiB multipart chunks**; verified parts with ETags and used **owner-scoped fingerprints** to detect duplicates and resume interrupted transfers.
- Decoupled uploads from video transcoding through an **SQS-to-Lambda-to-ECS/Fargate pipeline**, launching a **dedicated container task per video** to process source files asynchronously.
- Made queued processing **retry-safe** with **atomic MongoDB job claims**, preventing concurrent workers from processing the same video when queue messages are retried.
- Generated **adaptive-bitrate HLS** with **ffprobe and FFmpeg**, producing eligible **360p-1080p renditions**, six-second segments, and variant/master playlists in S3 for playback at multiple quality levels.
- Served private HLS assets through **CloudFront and S3 Origin Access Control**, integrating **Video.js/VHS adaptive quality switching** without routing each segment request through the application API.
- Codified the processing and delivery infrastructure with **SST**, including Lambda APIs, private S3 storage, SQS, **ECS autoscaling**, IAM and cache policies, and **Docker-based FFmpeg workers**.

SKILLS

Programming: TypeScript, JavaScript | **Backend & Data:** Node.js, Hono, MongoDB, PostgreSQL, Redis, REST APIs
Frontend: React, Axios, TanStack Query, Tailwind CSS
Cloud & DevOps: AWS, Docker, SST | **Developer Tools:** Git, GitHub, Turborepo