DEVX360

Sipho Sehlapelo
Kesley Hamann
Sibusiso Mngomezulu
Lwando Msindo
David Musa-Aisien

DevX360

INTRODUCTION

This document describes the Deployment Model for the DevX360 platform, outlining how the system's components are packaged, hosted, and executed in the target environment. The deployment model ensures that the frontend, API, and backend services are reliably delivered to end-users, with clear communication channels and well-defined responsibilities across all layers.

DEPLOYMENT MODEL

DevX360 is delivered as a cloud-hosted web application. The user-facing frontend is a static single-page app, and the server-side components (API and AI integration) run as managed services. Persistent data is stored in MongoDB Atlas and generated files are stored in object storage. We are currently considering using AWS hosting services for majority of our needs for its proven reliability and global availability, a broad portfolio of managed services and strong built-in security and compliance controls.

MAPPING OF LAYERS

- Frontend -- static site (S3 + CloudFront/CDN).
- Backend / API -- containerized service (ECS Fargate or simple app service).
- AI Analysis -- external API (OpenAI) called by the backend.
- Database -- managed cloud DB (MongoDB Atlas).
- Files/Reports -- object storage (S3).

JUSTIFICATION FOR THIS MODEL

- The frontend is a static React app, so hosting it as static files is cheap, fast and simple to operate.
- The API and AI logic are hosted as managed services to keep operational overhead low and allow straightforward scaling.
- Using a managed database (MongoDB Atlas) and a third-party AI service (OpenAI) removes the need to host and maintain DB or model infrastructure, simplifying security and maintenance.
- This approach keeps the architecture easy to explain, cost-effective for small deployments, and simple to scale if usage grows.

SECURITY NOTES

- TLS/HTTPS for all public endpoints.
- Secrets and API keys are stored securely (environment variables/secrets manager).

DIAGRAM