6. DevOps Implementation

Version Control Practices

Git Workflow

GitHub Integration

Continuous Integration/Continuous Deployment

GitHub Actions Configuration

Build Server Setup

Testing Pipeline

Containerization

Docker Implementation

Container Registry

Cloud Deployment

Google Cloud Setup

Netlify Frontend Deployment In our project, we utilize Netlify as the platform for hosting and deploying our frontend application. Netlify offers an automated and seamless deployment workflow by integrating directly with our GitHub repository. Every time changes are pushed to the main branch, Netlify automatically pulls the latest code, runs the specified build command (npm run build), and deploys the application to its globally distributed Content Delivery Network (CDN).

As part of this automated process, Netlify provides real-time feedback on the deployment status. If the npm run build command succeeds, the application is immediately deployed, and the live site is updated. If the build fails, Netlify generates detailed logs, making it easy to identify and resolve any issues. After deployment, Netlify confirms whether the site was successfully deployed, ensuring full transparency in the deployment process.

In addition, for every pull request, Netlify runs a Lighthouse audit to evaluate the performance, accessibility, best practices, and SEO of the site. This feature provides a clear score and actionable insights, helping us continuously optimize

the application before merging changes into the main branch. By catching potential issues at the pull request level, we ensure that only high-quality updates make it into production.

We have also implemented GitHub Actions as part of our CI/CD pipeline. These actions complement Netlify by automating additional checks, such as running unit tests, linting, and verifying configurations before deployment. GitHub Actions provide an extra layer of validation, ensuring that the codebase adheres to predefined quality standards and that potential issues are identified early in the development cycle.

Netlify's integration with GitHub and the use of Github Actions significantly streamline our deployment process. Features like automatic builds, deployment status notifications, and Lighthouse audits on pull requests, we are able to maintain a high-quality frontend application. Its global CDN ensures that the deployed site loads quickly for users worldwide, further enhancing the user experience. By leveraging Netlify, we have a reliable, efficient, and developer-friendly solution for managing our frontend deployment pipeline.

Another of the key advantages of using Netlify is that it offers its core features for free, including automated builds, deployments, and a globally distributed CDN. This makes it an excellent choice for small teams or projects, allowing us to deploy a high-quality frontend application without incurring additional costs.

Monitoring and Maintenance A key principle in DevOps is the ability to act based on metrics. To do this we use Micrometer metrics, a common monitoring facade which is vendor neutral, alike SLF4J is for logging. We use the Prometheus format as Quarkus provides a convenient library for combining these, `quarkus-micrometer-registry-prometheus`. To display and view these metrics we use Dashbuilder which allows easy visualization of the metric data via a YML format. Quarkus also has minimal config library for this, named `quarkus-dashbuilder`.

We have also looked into setting up a centralized logging system, but due to time constraints were unable to implement it in time for the writing of this report.

Another clear improvement is monitoring if our frontend server is up or overloaded.