DevOps Summary Report for Project Manager App

Date: August 07, 2025

Project Structure

- Frontend: React (Vite) app deployed on Vercel
- Backend: Node.js/Express API deployed on Render

Frontend Deployment (Vercel)

- Initialized Vite + React project
- Ensured project had `package.json`, `vite.config.ts`, and correct structure
- Set Vercel build settings:
- Build Command: `npm run build`
- Output Directory: `dist`
- Fixed missing `tsconfig.app.json` error
- Connected GitHub repo to Vercel for CI/CD
- Vercel auto-deploys on every push to main branch

Backend Deployment (Render)

- Deployed backend (Node.js/Express API) on Render
- Setup auto-deployment from GitHub repo
- Environment variables configured in Render dashboard
- Ensured server starts using: `npm run start` or `node index.js`

Environment Configuration

- Used `.env` files locally
- Configured production environment variables in:
- Render (for backend)
- Vercel (for frontend)

API Integration

- Frontend fetch calls point to Render-deployed server URL
- Used absolute path aliases (`@/`) configured via `vite.config.ts`

Domain Setup

- Set custom domain for frontend via Vercel dashboard (optional)
- Backend accessible via Render’s provided subdomain

Version Control

- Used Git and GitHub for source control
- Both Vercel and Render linked to GitHub repos for CI/CD pipelines

DevOps Principles Followed

- Continuous Integration/Continuous Deployment (CI/CD)
- Environment isolation (local vs production via `.env`)
- Cloud infrastructure provisioning (Vercel + Render)
- Monitoring via Vercel/Render dashboards

Automated Testing & CI

- Installed Jest and Supertest for backend API testing
- Created an example test file in `server/tests/auth.test.js`
- Added a `test` script to `server/package.json`
- Set up GitHub Actions workflow at `.github/workflows/test.yml` to run tests on every push and pull request

Error Monitoring

- Integrated Sentry for backend (Node.js) and frontend (React)
- Sentry initialized in backend (`server/index.js`) and frontend (`src/main.tsx`)
- Added `VITE_SENTRY_DSN` to `.env` for frontend and `SENTRY_DSN` for backend

Backend Security

- Added Helmet middleware for HTTP header security
- Added express-rate-limit to prevent brute-force attacks
- Added input validation and sanitization using express-validator in backend auth routes

Structured Logging

- Set up Winston logger in backend (`server/logger.js`)
- All requests and important events are logged

Performance & Bundle Analysis

- Installed `source-map-explorer` for frontend bundle analysis
- To analyze bundle: build frontend and run `npx source-map-explorer dist/assets/index-*.js`

Containerization

- Added a `Dockerfile` for backend to enable containerized deployments

Uptime Monitoring

- Backend URL added to UptimeRobot for downtime alerts

Additional Recommendations

- Expand backend and frontend tests for better coverage
- Use Sentry dashboards to monitor errors in production
- Regularly review and update dependencies to reduce vulnerabilities