1 3.6.3 Architectural Quality Requirements

The following prioritized quality attributes have been identified for the Coffee Shop Manager system, with testable and quantifiable metrics:

Priority	**Quality Requirement**	**Testable Metric / Specification**
1	Security	JWT-based authentication, hashed passwords, role-based access control; security audits quarterly with zero critical vulnerabilities
2	Usability	Responsive UI on mobile and desktop; >90% positive user feedback on usability surveys; accessibility compliance WCAG 2.1 AA level
3	Availability	System uptime $\geq$ 99.9% measured monthly; maximum downtime per month $\leq$ 43.2 minutes
4	Performance	API average response time < 300ms under typical load of 100 concurrent users
5	Testability	Critical modules covered by automated tests with $\geq$ 90% unit test coverage

Table 1: Prioritized Architectural Quality Requirements

2 3.6.5 Architectural Constraints

The system architecture is influenced by the following constraints:

- The database must be implemented using PostgreSQL.
- The mobile application must be developed using React Native.
- The backend must use Node.js with Express and be deployable via Docker containers.
- Only open-source libraries and frameworks may be used.
- Mobile and web clients must share the same backend API.
- The web frontend must support the latest versions of Chrome, Edge, Safari, and Firefox.
- Authentication must utilize JWT with role-based access control.
- The system must be deployable on cloud platforms that support Docker.

3 3.6.6 Technology Choices

Below is an assessment of major system components with options, pros and cons, and final justification.

Technology	**Overview**	**Pros**	**Cons**	**Justification**
Next.js	React framework with SSR and SSG	Great SEO, fast performance, strong React ecosystem	Slightly steeper learning curve	Fits requirement for modern, performant, SEO-friendly web app
React SPA	Client-only React app	Simple to set up, fast developer iteration	Less SEO friendly	SEO needed, so Next.js preferred
Angular	Full-featured frontend framework	Rich features, powerful tooling	More complex, larger bundle size	Overkill for this app; React ecosystem preferred

Table 2: Web Frontend Technology Options

3.1 Web Frontend

3.2 Mobile App

Technology	**Overview**	**Pros**	**Cons**	**Justification**
React Native	Cross-platform mobile framework	Single codebase iOS/Android, good community	Native modules sometimes needed	Required by constraints; fits well with React frontend
Flutter	Google's UI toolkit for mobile	Fast performance, great UI	Different language (Dart), disallowed by constraints	Disqualified by constraints
Native iOS/Android	Platform-specific apps	Best performance and UX	Higher cost and development time	Not feasible due to resource and time constraints

Table 3: Mobile App Technology Options

3.3 Backend API

3.4 Database

3.5 Authentication

Technology	**Overview**	**Pros**	**Cons**	**Justification**
Node.js + Express	JavaScript runtime + minimalist web framework	Fast, scalable, large community, JS sharing with frontend	Callback complexity mitigated by async/await	Matches project constraints and developer skills
Django (Python)	Full-featured web framework	Batteries included, secure by default	Different language, not aligned with frontend tech	Not chosen due to JS ecosystem preference
Spring Boot (Java)	Enterprise-level framework	Robust, scalable	Heavyweight, longer development time	Overkill for this project

Table 4: Backend API Technology Options

Technology	**Overview**	**Pros**	**Cons**	**Justification**
PostgreSQL + Drizzle ORM	Relational DB + modern ORM	ACID compliance, strong features, TypeScript ORM support	Complex horizontal scaling	Required by constraints; strong data integrity needed
MongoDB	NoSQL document store	Flexible schema, rapid iteration	Less suited for complex transactions	Schema complexity not a good fit
MySQL	Popular relational DB	Widely supported, good performance	Licensing on some editions	PostgreSQL preferred due to features

Table 5: Database Technology Options

Technology	**Overview**	**Pros**	**Cons**	**Justification**
JWT + Role-Based Access Control	Token-based auth with permission control	Stateless, scalable, flexible	Need careful implementation to avoid token leaks	Standard approach, fits security requirements
OAuth 2.0	Industry standard delegated auth	Secure, widely adopted	More complex setup	More than needed for this app
Session Cookies	Traditional server sessions	Simple, secure when properly implemented	Scalability issues	Less scalable than JWT

Table 6: Authentication Technology Options