# Complete Coding Prompts Library - All Templates

\*Comprehensive collection of all coding prompts and templates\*

\*Optimized for lovable.dev (Frontend) and Windsurf (Backend) platforms\*

---

## Table of Contents

1. [Business Strategy Templates](#business-strategy-templates)

2. [Design & Planning Templates](#design--planning-templates)

3. [UI/UX Design Templates](#uiux-design-templates)

4. [Frontend Development Templates](#frontend-development-templates)

5. [Backend Development Templates](#backend-development-templates)

6. [Full-Stack Integration Templates](#full-stack-integration-templates)

7. [DevOps & Deployment Templates](#devops--deployment-templates)

8. [Code Quality & Maintenance Templates](#code-quality--maintenance-templates)

9. [Collaboration & Project Management Templates](#collaboration--project-management-templates)

10. [AI-Enhanced Application Features](#ai-enhanced-application-features)

11. [Advanced Integration & Specialized Workflows](#advanced-integration--specialized-workflows)

---

## Business Strategy Templates

### 1. Comprehensive Business Requirements Document (BRD) Template

```

Act as a business analyst creating a comprehensive Business Requirements Document. Document requirements for [PROJECT\_NAME]:

\*\*Executive Summary:\*\*

- Project Overview: [PROJECT\_OVERVIEW\_DESCRIPTION]

- Business Justification: [BUSINESS\_JUSTIFICATION]

- Expected Benefits: [EXPECTED\_BENEFITS]

- Success Metrics: [SUCCESS\_METRICS\_DEFINITION]

- Investment Summary: [INVESTMENT\_SUMMARY]

\*\*Business Context:\*\*

- Current State Analysis: [CURRENT\_STATE\_DESCRIPTION]

- Business Problems: [BUSINESS\_PROBLEMS\_IDENTIFICATION]

- Opportunity Assessment: [OPPORTUNITY\_ASSESSMENT]

- Market Context: [MARKET\_CONTEXT\_ANALYSIS]

- Competitive Landscape: [COMPETITIVE\_LANDSCAPE\_ANALYSIS]

\*\*Stakeholder Analysis:\*\*

- Primary Stakeholders: [PRIMARY\_STAKEHOLDERS\_IDENTIFICATION]

- Secondary Stakeholders: [SECONDARY\_STAKEHOLDERS\_IDENTIFICATION]

- Decision Makers: [DECISION\_MAKERS\_MAPPING]

- Influence/Interest Matrix: [STAKEHOLDER\_INFLUENCE\_MATRIX]

- Communication Requirements: [STAKEHOLDER\_COMMUNICATION\_REQUIREMENTS]

\*\*Business Requirements:\*\*

- Functional Requirements: [FUNCTIONAL\_REQUIREMENTS\_SPECIFICATION]

- Non-Functional Requirements: [NON\_FUNCTIONAL\_REQUIREMENTS\_SPECIFICATION]

- Performance Requirements: [PERFORMANCE\_REQUIREMENTS\_DEFINITION]

- Security Requirements: [SECURITY\_REQUIREMENTS\_SPECIFICATION]

- Compliance Requirements: [COMPLIANCE\_REQUIREMENTS\_DEFINITION]

\*\*User Requirements:\*\*

- User Personas: [USER\_PERSONAS\_DEFINITION]

- User Stories: [USER\_STORIES\_DOCUMENTATION]

- User Journeys: [USER\_JOURNEY\_MAPPING]

- Use Cases: [USE\_CASES\_SPECIFICATION]

- Acceptance Criteria: [ACCEPTANCE\_CRITERIA\_DEFINITION]

\*\*Success Criteria:\*\*

- Quantitative Metrics: [QUANTITATIVE\_SUCCESS\_METRICS]

- Qualitative Metrics: [QUALITATIVE\_SUCCESS\_METRICS]

- Key Performance Indicators: [KPI\_DEFINITION]

- Success Validation: [SUCCESS\_VALIDATION\_APPROACH]

- ROI Measurement: [ROI\_MEASUREMENT\_FRAMEWORK]

Provide comprehensive business requirements documentation with stakeholder analysis, detailed requirements specifications, and success measurement frameworks.

```

### 2. Statement of Work (SOW) Template

```

Act as a project manager creating a detailed Statement of Work. Create SOW for [PROJECT\_NAME] with 10 full stack developers:

\*\*Project Overview:\*\*

- Project Description: [PROJECT\_DESCRIPTION]

- Project Objectives: [PROJECT\_OBJECTIVES\_DEFINITION]

- Business Justification: [BUSINESS\_JUSTIFICATION]

- Success Criteria: [SUCCESS\_CRITERIA\_SPECIFICATION]

- Project Value: [PROJECT\_VALUE\_PROPOSITION]

\*\*Scope of Work:\*\*

- Included Services: [INCLUDED\_SERVICES\_SPECIFICATION]

- Deliverables: [DELIVERABLES\_DEFINITION]

- Service Boundaries: [SERVICE\_BOUNDARIES\_CLARIFICATION]

- Exclusions: [EXCLUSIONS\_SPECIFICATION]

- Change Management: [CHANGE\_MANAGEMENT\_PROCESS]

\*\*Project Deliverables:\*\*

- Phase 1 Deliverables: [PHASE\_1\_DELIVERABLES\_SPECIFICATION]

- Phase 2 Deliverables: [PHASE\_2\_DELIVERABLES\_SPECIFICATION]

- Phase 3 Deliverables: [PHASE\_3\_DELIVERABLES\_SPECIFICATION]

- Final Deliverables: [FINAL\_DELIVERABLES\_SPECIFICATION]

- Acceptance Criteria: [DELIVERABLE\_ACCEPTANCE\_CRITERIA]

\*\*Technical Specifications:\*\*

- Technology Stack: [TECHNOLOGY\_STACK\_SPECIFICATION]

- Architecture Requirements: [ARCHITECTURE\_REQUIREMENTS\_DEFINITION]

- Performance Requirements: [PERFORMANCE\_REQUIREMENTS\_SPECIFICATION]

- Security Requirements: [SECURITY\_REQUIREMENTS\_DEFINITION]

- Integration Requirements: [INTEGRATION\_REQUIREMENTS\_SPECIFICATION]

\*\*Project Timeline:\*\*

- Project Duration: [PROJECT\_DURATION\_ESTIMATION]

- Milestone Schedule: [MILESTONE\_SCHEDULE\_DEFINITION]

- Phase Timelines: [PHASE\_TIMELINES\_SPECIFICATION]

- Critical Path: [CRITICAL\_PATH\_IDENTIFICATION]

- Dependencies: [DEPENDENCIES\_MANAGEMENT]

\*\*Resource Allocation:\*\*

- Team Structure: [TEAM\_STRUCTURE\_DEFINITION]

- Role Assignments: [ROLE\_ASSIGNMENTS\_SPECIFICATION]

- Resource Availability: [RESOURCE\_AVAILABILITY\_PLANNING]

- Skill Requirements: [SKILL\_REQUIREMENTS\_DEFINITION]

- Resource Scalability: [RESOURCE\_SCALABILITY\_PLANNING]

Provide complete statement of work with technical specifications, resource planning, timeline management, and delivery frameworks.

```

### 3. Project Charter Template

```

Act as a project manager creating a comprehensive project charter. Create project charter for [PROJECT\_NAME]:

\*\*Project Information:\*\*

- Project Name: [PROJECT\_NAME]

- Project Manager: [PROJECT\_MANAGER\_NAME]

- Project Sponsor: [PROJECT\_SPONSOR\_IDENTIFICATION]

- Start Date: [PROJECT\_START\_DATE]

- Target Completion: [PROJECT\_TARGET\_COMPLETION]

\*\*Project Overview:\*\*

- Project Description: [PROJECT\_DESCRIPTION]

- Business Case: [BUSINESS\_CASE\_JUSTIFICATION]

- Project Purpose: [PROJECT\_PURPOSE\_STATEMENT]

- Success Criteria: [SUCCESS\_CRITERIA\_DEFINITION]

- Key Benefits: [KEY\_BENEFITS\_IDENTIFICATION]

\*\*Project Objectives:\*\*

- Primary Objectives: [PRIMARY\_OBJECTIVES\_DEFINITION]

- Secondary Objectives: [SECONDARY\_OBJECTIVES\_SPECIFICATION]

- Measurable Outcomes: [MEASURABLE\_OUTCOMES\_DEFINITION]

- Success Metrics: [SUCCESS\_METRICS\_SPECIFICATION]

- Performance Indicators: [PERFORMANCE\_INDICATORS\_ESTABLISHMENT]

\*\*Project Scope:\*\*

- Project Boundaries: [PROJECT\_BOUNDARIES\_DEFINITION]

- Included Deliverables: [INCLUDED\_DELIVERABLES\_SPECIFICATION]

- Excluded Items: [EXCLUDED\_ITEMS\_CLARIFICATION]

- Project Constraints: [PROJECT\_CONSTRAINTS\_IDENTIFICATION]

- Project Assumptions: [PROJECT\_ASSUMPTIONS\_DOCUMENTATION]

\*\*Stakeholder Information:\*\*

- Project Sponsor: [PROJECT\_SPONSOR\_DETAILS]

- Steering Committee: [STEERING\_COMMITTEE\_COMPOSITION]

- Key Stakeholders: [KEY\_STAKEHOLDERS\_IDENTIFICATION]

- Project Team: [PROJECT\_TEAM\_COMPOSITION]

- End Users: [END\_USER\_IDENTIFICATION]

Provide comprehensive project charter with governance structure, scope definition, and stakeholder management framework.

```

---

## Design & Planning Templates

### 4. System Architecture Design

```

Act as a senior software architect designing a comprehensive system architecture. Create system architecture for [PROJECT\_NAME] that:

\*\*Architecture Overview:\*\*

- System Purpose: [SYSTEM\_PURPOSE\_AND\_GOALS]

- Architecture Style: [MONOLITHIC/MICROSERVICES/SERVERLESS/HYBRID]

- Technology Stack: [FRONTEND\_TECH/BACKEND\_TECH/DATABASE\_TECH/INFRASTRUCTURE]

- Deployment Model: [CLOUD/ON\_PREMISE/HYBRID/MULTI\_CLOUD]

- Scalability Requirements: [EXPECTED\_LOAD/GROWTH\_PROJECTIONS/PERFORMANCE\_TARGETS]

\*\*Component Design:\*\*

- Frontend Components: [UI\_FRAMEWORK/STATE\_MANAGEMENT/ROUTING/BUILD\_TOOLS]

- Backend Services: [API\_LAYER/BUSINESS\_LOGIC/DATA\_ACCESS/AUTHENTICATION]

- Database Architecture: [DATABASE\_TYPE/SCHEMA\_DESIGN/INDEXING\_STRATEGY/BACKUP\_RECOVERY]

- External Integrations: [THIRD\_PARTY\_APIS/PAYMENT\_SYSTEMS/ANALYTICS/MONITORING]

- Infrastructure Components: [LOAD\_BALANCERS/CDN/CACHING/MESSAGE\_QUEUES]

\*\*Security Architecture:\*\*

- Authentication Strategy: [OAUTH/JWT/SAML/MULTI\_FACTOR]

- Authorization Model: [RBAC/ABAC/CUSTOM\_PERMISSIONS]

- Data Protection: [ENCRYPTION\_AT\_REST/IN\_TRANSIT/KEY\_MANAGEMENT]

- Network Security: [FIREWALLS/VPN/API\_GATEWAY/RATE\_LIMITING]

- Compliance Requirements: [GDPR/HIPAA/SOC2/CUSTOM\_COMPLIANCE]

\*\*Performance & Scalability:\*\*

- Caching Strategy: [BROWSER\_CACHE/CDN/DATABASE\_CACHE/APPLICATION\_CACHE]

- Load Distribution: [HORIZONTAL\_SCALING/VERTICAL\_SCALING/AUTO\_SCALING]

- Database Optimization: [QUERY\_OPTIMIZATION/INDEXING/PARTITIONING/REPLICATION]

- Content Delivery: [CDN\_STRATEGY/ASSET\_OPTIMIZATION/LAZY\_LOADING]

- Monitoring & Observability: [LOGGING/METRICS/TRACING/ALERTING]

\*\*Integration Architecture:\*\*

- API Design: [REST/GRAPHQL/WEBSOCKETS/MESSAGE\_QUEUES]

- Data Flow: [DATA\_INGESTION/PROCESSING/STORAGE/RETRIEVAL]

- Event Architecture: [EVENT\_SOURCING/CQRS/MESSAGE\_BROKERS]

- Third-party Integration: [WEBHOOK\_HANDLING/API\_AGGREGATION/DATA\_SYNCHRONIZATION]

For lovable.dev frontend implementation:

- Component architecture with reusable UI components

- State management strategy (Context/Redux/Zustand)

- Responsive design system integration

- AWS Amplify deployment optimization

For windsurf backend implementation:

- Microservices architecture with clear service boundaries

- Database design with optimization for cost and performance

- API gateway configuration for security and routing

- AWS services integration for scalability and reliability

Provide complete system architecture with component diagrams, data flow diagrams, deployment architecture, and implementation guidelines.

```

### 5. Database Schema Design

```

Act as a database architect designing a comprehensive database schema. Create database schema for [PROJECT\_NAME] that:

\*\*Database Strategy:\*\*

- Database Type: [RELATIONAL/NOSQL/GRAPH/TIME\_SERIES/HYBRID]

- Technology Choice: [POSTGRESQL/MYSQL/MONGODB/DYNAMODB/CUSTOM]

- Architecture Pattern: [SINGLE\_DATABASE/MICROSERVICES\_DATABASES/DATABASE\_PER\_SERVICE]

- Deployment Model: [CLOUD\_MANAGED/SELF\_HOSTED/HYBRID]

- Backup & Recovery: [AUTOMATED\_BACKUPS/POINT\_IN\_TIME\_RECOVERY/DISASTER\_RECOVERY]

\*\*Schema Design:\*\*

- Entity Relationships: [PRIMARY\_ENTITIES/RELATIONSHIPS/CARDINALITY]

- Table Structure: [TABLE\_DEFINITIONS/COLUMN\_SPECIFICATIONS/DATA\_TYPES]

- Indexing Strategy: [PRIMARY\_INDEXES/SECONDARY\_INDEXES/COMPOSITE\_INDEXES]

- Constraints: [FOREIGN\_KEYS/UNIQUE\_CONSTRAINTS/CHECK\_CONSTRAINTS]

- Normalization: [NORMALIZATION\_LEVEL/DENORMALIZATION\_DECISIONS]

\*\*Performance Optimization:\*\*

- Query Optimization: [QUERY\_PATTERNS/INDEX\_OPTIMIZATION/EXECUTION\_PLANS]

- Caching Strategy: [QUERY\_CACHING/RESULT\_CACHING/CONNECTION\_POOLING]

- Partitioning: [HORIZONTAL\_PARTITIONING/VERTICAL\_PARTITIONING/SHARDING]

- Read Replicas: [READ\_SCALING/EVENTUAL\_CONSISTENCY/FAILOVER]

- Connection Management: [CONNECTION\_POOLING/CONNECTION\_LIMITS/LOAD\_BALANCING]

\*\*Data Security:\*\*

- Access Control: [USER\_ROLES/PERMISSIONS/ROW\_LEVEL\_SECURITY]

- Encryption: [ENCRYPTION\_AT\_REST/ENCRYPTION\_IN\_TRANSIT/COLUMN\_ENCRYPTION]

- Auditing: [ACCESS\_LOGGING/CHANGE\_TRACKING/COMPLIANCE\_REPORTING]

- Data Masking: [SENSITIVE\_DATA\_HANDLING/PII\_PROTECTION/ANONYMIZATION]

- Backup Security: [ENCRYPTED\_BACKUPS/SECURE\_STORAGE/ACCESS\_CONTROLS]

\*\*Scalability & Maintenance:\*\*

- Growth Planning: [CAPACITY\_PLANNING/STORAGE\_GROWTH/PERFORMANCE\_SCALING]

- Migration Strategy: [SCHEMA\_MIGRATIONS/DATA\_MIGRATIONS/VERSION\_CONTROL]

- Monitoring: [PERFORMANCE\_MONITORING/HEALTH\_CHECKS/ALERTING]

- Maintenance: [REGULAR\_MAINTENANCE/INDEX\_MAINTENANCE/STATISTICS\_UPDATES]

For windsurf backend optimization:

- Cost-effective AWS RDS/DynamoDB configuration

- Query optimization for minimal compute usage

- Efficient data modeling for API performance

- Automated backup and disaster recovery setup

Provide complete database schema with entity relationship diagrams, table definitions, indexing strategies, and migration scripts.

```

### 6. API Design Blueprint

```

Act as an API architect designing a comprehensive API strategy. Create API design for [PROJECT\_NAME] that:

\*\*API Architecture:\*\*

- API Style: [REST/GRAPHQL/GRPC/HYBRID]

- Protocol: [HTTP/HTTPS/WEBSOCKETS/MESSAGE\_QUEUES]

- Architecture Pattern: [MONOLITHIC\_API/MICROSERVICES\_APIS/BFF\_PATTERN]

- Gateway Strategy: [API\_GATEWAY/SERVICE\_MESH/DIRECT\_ACCESS]

- Versioning Strategy: [URL\_VERSIONING/HEADER\_VERSIONING/CONTENT\_NEGOTIATION]

\*\*API Design Standards:\*\*

- Resource Modeling: [RESOURCE\_IDENTIFICATION/HIERARCHY/RELATIONSHIPS]

- URL Structure: [NAMING\_CONVENTIONS/PATH\_PARAMETERS/QUERY\_PARAMETERS]

- HTTP Methods: [GET/POST/PUT/DELETE/PATCH\_USAGE]

- Status Codes: [SUCCESS\_CODES/ERROR\_CODES/CUSTOM\_CODES]

- Headers: [STANDARD\_HEADERS/CUSTOM\_HEADERS/SECURITY\_HEADERS]

\*\*Request/Response Design:\*\*

- Data Formats: [JSON/XML/PROTOBUF/CUSTOM\_FORMATS]

- Request Validation: [SCHEMA\_VALIDATION/INPUT\_SANITIZATION/TYPE\_CHECKING]

- Response Structure: [CONSISTENT\_FORMAT/ERROR\_HANDLING/PAGINATION]

- Content Negotiation: [ACCEPT\_HEADERS/CONTENT\_TYPE/COMPRESSION]

- Filtering & Searching: [QUERY\_PARAMETERS/SEARCH\_SYNTAX/SORTING]

\*\*Security Implementation:\*\*

- Authentication: [OAUTH2/JWT/API\_KEYS/MUTUAL\_TLS]

- Authorization: [ROLE\_BASED/RESOURCE\_BASED/SCOPE\_BASED]

- Rate Limiting: [REQUEST\_LIMITS/USER\_LIMITS/ENDPOINT\_LIMITS]

- Input Validation: [PARAMETER\_VALIDATION/PAYLOAD\_VALIDATION/SQL\_INJECTION\_PREVENTION]

- Security Headers: [CORS/CSP/SECURITY\_HEADERS]

\*\*Performance & Scalability:\*\*

- Caching: [RESPONSE\_CACHING/EDGE\_CACHING/CONDITIONAL\_REQUESTS]

- Pagination: [OFFSET\_PAGINATION/CURSOR\_PAGINATION/LIMIT\_CONTROLS]

- Compression: [GZIP/BROTLI/RESPONSE\_COMPRESSION]

- Connection Management: [KEEP\_ALIVE/CONNECTION\_POOLING/TIMEOUT\_HANDLING]

- Load Balancing: [ROUND\_ROBIN/LEAST\_CONNECTIONS/HEALTH\_CHECKS]

\*\*Error Handling:\*\*

- Error Response Format: [CONSISTENT\_ERROR\_STRUCTURE/ERROR\_CODES/MESSAGES]

- Error Categories: [CLIENT\_ERRORS/SERVER\_ERRORS/BUSINESS\_LOGIC\_ERRORS]

- Error Details: [FIELD\_VALIDATION\_ERRORS/CONSTRAINT\_VIOLATIONS/SUGGESTIONS]

- Logging: [ERROR\_LOGGING/REQUEST\_LOGGING/AUDIT\_TRAILS]

- Monitoring: [ERROR\_RATE\_MONITORING/PERFORMANCE\_MONITORING/ALERTING]

\*\*Documentation & Testing:\*\*

- API Documentation: [OPENAPI\_SPEC/SWAGGER/POSTMAN\_COLLECTIONS]

- Interactive Documentation: [SWAGGER\_UI/REDOC/CUSTOM\_PORTALS]

- Code Examples: [MULTIPLE\_LANGUAGES/SDK\_GENERATION/SAMPLE\_REQUESTS]

- Testing Strategy: [UNIT\_TESTS/INTEGRATION\_TESTS/CONTRACT\_TESTS]

- Mocking: [DEVELOPMENT\_MOCKS/TESTING\_MOCKS/SANDBOX\_ENVIRONMENTS]

For lovable.dev frontend integration:

- Frontend-optimized API endpoints

- Real-time updates via WebSockets/SSE

- Efficient data fetching patterns

- Error handling for UI components

For windsurf backend implementation:

- Cost-optimized API design for minimal compute usage

- Efficient database query patterns

- AWS Lambda/Container integration

- Monitoring and logging for production

Provide complete API specification with OpenAPI documentation, implementation examples, security guidelines, and testing strategies.

```

---

## UI/UX Design Templates

### 7. Responsive UI Component Design

```

Act as a senior UI/UX designer specializing in lovable.dev implementations. Design a responsive UI component for [COMPONENT\_NAME] that:

\*\*Component Requirements:\*\*

- Component Type: [BUTTON/FORM/CARD/NAVIGATION/DATA\_DISPLAY/CUSTOM]

- Primary Function: [USER\_INTERACTION\_PURPOSE]

- Brand Integration: [BRAND\_COLORS/TYPOGRAPHY/VISUAL\_STYLE]

- Accessibility Level: [WCAG\_AA/WCAG\_AAA/CUSTOM\_REQUIREMENTS]

- Device Support: [MOBILE/TABLET/DESKTOP/ALL\_DEVICES]

\*\*Responsive Design:\*\*

- Breakpoints: [MOBILE\_320PX/TABLET\_768PX/DESKTOP\_1024PX/LARGE\_1440PX]

- Layout Adaptation: [COMPONENT\_SIZING/CONTENT\_REFLOW/INTERACTION\_CHANGES]

- Touch Optimization: [TOUCH\_TARGETS/GESTURE\_SUPPORT/MOBILE\_INTERACTIONS]

- Content Prioritization: [MOBILE\_FIRST\_CONTENT/PROGRESSIVE\_DISCLOSURE]

- Performance Considerations: [LAZY\_LOADING/EFFICIENT\_RENDERING/RESOURCE\_OPTIMIZATION]

\*\*Visual Design:\*\*

- Color System: [PRIMARY\_COLORS/SECONDARY\_COLORS/STATE\_COLORS/ACCESSIBILITY\_CONTRAST]

- Typography: [FONT\_FAMILIES/FONT\_SIZES/LINE\_HEIGHTS/RESPONSIVE\_SCALING]

- Spacing System: [PADDING/MARGINS/GRID\_SYSTEM/RESPONSIVE\_SPACING]

- Visual Hierarchy: [EMPHASIS/CONTRAST/ALIGNMENT/PROXIMITY]

- Brand Consistency: [LOGO\_INTEGRATION/BRAND\_VOICE/VISUAL\_IDENTITY]

\*\*Interaction Design:\*\*

- Interactive States: [DEFAULT/HOVER/ACTIVE/FOCUS/DISABLED/LOADING]

- Micro-interactions: [BUTTON\_ANIMATIONS/FORM\_FEEDBACK/TRANSITION\_EFFECTS]

- User Feedback: [VISUAL\_FEEDBACK/HAPTIC\_FEEDBACK/AUDIO\_FEEDBACK]

- Error Handling: [ERROR\_STATES/VALIDATION\_FEEDBACK/RECOVERY\_ACTIONS]

- Accessibility: [KEYBOARD\_NAVIGATION/SCREEN\_READER\_SUPPORT/HIGH\_CONTRAST]

\*\*Component Architecture:\*\*

- Props Interface: [REQUIRED\_PROPS/OPTIONAL\_PROPS/DEFAULT\_VALUES]

- Variants: [SIZE\_VARIANTS/COLOR\_VARIANTS/STYLE\_VARIANTS]

- Composition: [CHILD\_COMPONENTS/SLOT\_SYSTEM/FLEXIBLE\_CONTENT]

- Theming: [THEME\_INTEGRATION/CUSTOM\_STYLING/CSS\_VARIABLES]

- Reusability: [COMPONENT\_LIBRARY\_INTEGRATION/DOCUMENTATION/EXAMPLES]

\*\*Technical Implementation:\*\*

- HTML Structure: [SEMANTIC\_MARKUP/ACCESSIBILITY\_ATTRIBUTES/PROPER\_NESTING]

- CSS Architecture: [COMPONENT\_STYLES/RESPONSIVE\_STYLES/UTILITY\_CLASSES]

- JavaScript Behavior: [EVENT\_HANDLING/STATE\_MANAGEMENT/PERFORMANCE\_OPTIMIZATION]

- Framework Integration: [REACT\_COMPONENT/VUE\_COMPONENT/VANILLA\_JS]

- Testing Strategy: [VISUAL\_TESTING/INTERACTION\_TESTING/ACCESSIBILITY\_TESTING]

\*\*AWS Amplify Integration:\*\*

- Build Optimization: [COMPONENT\_BUNDLING/TREE\_SHAKING/CODE\_SPLITTING]

- Asset Management: [IMAGE\_OPTIMIZATION/FONT\_LOADING/RESOURCE\_CACHING]

- Performance: [CORE\_WEB\_VITALS/LOADING\_OPTIMIZATION/RENDERING\_PERFORMANCE]

- Deployment: [CI\_CD\_INTEGRATION/PREVIEW\_DEPLOYMENTS/ROLLBACK\_STRATEGIES]

Provide complete component design with specifications, responsive behavior documentation, accessibility compliance, and implementation guidelines for lovable.dev.

```

### 8. User Experience Flow Design

```

Act as a UX flow specialist for lovable.dev implementations. Design a comprehensive user experience flow for [PROCESS\_NAME] that:

\*\*Flow Architecture:\*\*

- Flow Type: [LINEAR/BRANCHING/PARALLEL/ADAPTIVE]

- Entry Points: [DIRECT\_ACCESS/REFERRAL/SEARCH/MARKETING\_CAMPAIGN]

- User Goals: [PRIMARY\_OBJECTIVES/SECONDARY\_GOALS/SUCCESS\_METRICS]

- Flow Length: [SINGLE\_PAGE/MULTI\_STEP/PROGRESSIVE\_DISCLOSURE]

- Completion Time: [EXPECTED\_DURATION/TIME\_CONSTRAINTS/URGENCY\_FACTORS]

\*\*User Journey Mapping:\*\*

- User Personas: [PRIMARY\_USERS/SECONDARY\_USERS/EDGE\_CASE\_USERS]

- Touchpoints: [ENTRY\_POINTS/INTERACTION\_POINTS/DECISION\_POINTS/EXIT\_POINTS]

- Emotional Journey: [USER\_EMOTIONS/PAIN\_POINTS/DELIGHT\_MOMENTS/FRUSTRATION\_POINTS]

- Context Variations: [DEVICE\_CONTEXTS/TIME\_CONTEXTS/LOCATION\_CONTEXTS/MOOD\_CONTEXTS]

- Behavior Patterns: [EXPECTED\_BEHAVIORS/ACTUAL\_BEHAVIORS/DEVIATION\_HANDLING]

\*\*Step-by-Step Flow Design:\*\*

1. \*\*Entry & Orientation\*\*

- Landing Experience: [CLEAR\_PURPOSE/VALUE\_PROPOSITION/VISUAL\_HIERARCHY]

- Onboarding: [PROGRESSIVE\_DISCLOSURE/GUIDED\_TOUR/QUICK\_START]

- Expectation Setting: [PROGRESS\_INDICATORS/TIME\_ESTIMATES/REQUIRED\_INFORMATION]

2. \*\*Information Gathering\*\*

- Form Design: [LOGICAL\_GROUPING/PROGRESSIVE\_DISCLOSURE/SMART\_DEFAULTS]

- Validation: [REAL\_TIME\_VALIDATION/CLEAR\_ERROR\_MESSAGES/RECOVERY\_GUIDANCE]

- Data Input: [EFFICIENT\_INPUT\_METHODS/AUTO\_COMPLETION/FORMAT\_ASSISTANCE]

3. \*\*Decision Points\*\*

- Choice Architecture: [CLEAR\_OPTIONS/DEFAULT\_SELECTIONS/COMPARISON\_TOOLS]

- Information Support: [TOOLTIPS/HELP\_TEXT/DETAILED\_EXPLANATIONS]

- Confidence Building: [SOCIAL\_PROOF/TESTIMONIALS/GUARANTEE\_INFORMATION]

4. \*\*Confirmation & Completion\*\*

- Review Process: [SUMMARY\_DISPLAY/EDIT\_CAPABILITIES/CONFIRMATION\_STEPS]

- Final Actions: [CLEAR\_CALL\_TO\_ACTION/SINGLE\_CLICK\_COMPLETION/SUCCESS\_CONFIRMATION]

- Next Steps: [IMMEDIATE\_NEXT\_ACTIONS/FOLLOW\_UP\_COMMUNICATIONS/ONGOING\_ENGAGEMENT]

\*\*Error Prevention & Recovery:\*\*

- Input Validation: [REAL\_TIME\_FEEDBACK/FORMAT\_GUIDANCE/ERROR\_PREVENTION]

- Error Messages: [CLEAR\_LANGUAGE/SPECIFIC\_GUIDANCE/RECOVERY\_ACTIONS]

- Recovery Paths: [EASY\_CORRECTION/SAVE\_PROGRESS/ALTERNATIVE\_PATHS]

- Fallback Options: [ALTERNATIVE\_COMPLETION\_METHODS/SUPPORT\_CONTACT/SAVE\_FOR\_LATER]

\*\*Responsive & Accessibility:\*\*

- Mobile Optimization: [TOUCH\_FRIENDLY\_INTERACTIONS/SIMPLIFIED\_STEPS/SWIPE\_NAVIGATION]

- Accessibility Features: [KEYBOARD\_NAVIGATION/SCREEN\_READER\_SUPPORT/HIGH\_CONTRAST]

- Loading States: [PROGRESS\_INDICATORS/SKELETON\_SCREENS/PERFORMANCE\_OPTIMIZATION]

- Offline Capabilities: [OFFLINE\_SAVE/SYNC\_WHEN\_ONLINE/OFFLINE\_INDICATORS]

\*\*Conversion Optimization:\*\*

- Friction Reduction: [MINIMAL\_STEPS/SMART\_DEFAULTS/AUTO\_FILL\_OPTIONS]

- Trust Building: [SECURITY\_INDICATORS/PRIVACY\_STATEMENTS/TESTIMONIALS]

- Motivation Enhancement: [PROGRESS\_GAMIFICATION/SOCIAL\_PROOF/URGENCY\_INDICATORS]

- A/B Testing: [VARIANT\_OPPORTUNITIES/METRICS\_TRACKING/OPTIMIZATION\_AREAS]

\*\*Technical Integration:\*\*

- Frontend Implementation: [COMPONENT\_INTEGRATION/STATE\_MANAGEMENT/ROUTING]

- Backend Communication: [API\_CALLS/DATA\_VALIDATION/ERROR\_HANDLING]

- Analytics Integration: [EVENT\_TRACKING/FUNNEL\_ANALYSIS/CONVERSION\_MEASUREMENT]

- Performance: [FAST\_LOADING/SMOOTH\_TRANSITIONS/EFFICIENT\_RENDERING]

Provide complete user flow with wireframes, journey maps, decision trees, and implementation guidelines optimized for lovable.dev and user conversion.

```

### 9. Information Architecture Design

```

Act as an information architecture specialist for lovable.dev implementations. Design a comprehensive information architecture for [WEBSITE/APPLICATION] that:

\*\*Content Strategy:\*\*

- Content Types: [PAGES/ARTICLES/PRODUCTS/MEDIA/DOCUMENTS/USER\_GENERATED]

- Content Volume: [CURRENT\_CONTENT\_AMOUNT/GROWTH\_PROJECTIONS/CONTENT\_LIFECYCLE]

- Content Relationships: [HIERARCHICAL/NETWORK/FACETED/TAGGED\_RELATIONSHIPS]

- Content Management: [CREATION\_WORKFLOW/APPROVAL\_PROCESS/UPDATE\_FREQUENCY]

- Content Governance: [QUALITY\_STANDARDS/OWNERSHIP/MAINTENANCE\_RESPONSIBILITY]

\*\*Organizational Structure:\*\*

- Site Architecture: [HIERARCHICAL\_STRUCTURE/FLAT\_STRUCTURE/HYBRID\_APPROACH]

- Navigation Depth: [SHALLOW\_HIERARCHY/DEEP\_HIERARCHY/BALANCED\_APPROACH]

- Content Categorization: [TOPIC\_BASED/AUDIENCE\_BASED/TASK\_BASED/HYBRID]

- Taxonomy Design: [CATEGORIES/TAGS/METADATA/FACETED\_CLASSIFICATION]

- URL Structure: [LOGICAL\_HIERARCHY/SEO\_OPTIMIZATION/USER\_FRIENDLY\_URLS]

\*\*Navigation Design:\*\*

- Primary Navigation: [GLOBAL\_MENU/MAIN\_CATEGORIES/PRIORITY\_AREAS]

- Secondary Navigation: [SUB\_CATEGORIES/CONTEXTUAL\_MENUS/BREADCRUMBS]

- Utility Navigation: [ACCOUNT\_ACCESS/SEARCH/HELP/CONTACT]

- Mobile Navigation: [HAMBURGER\_MENU/BOTTOM\_NAV/DRAWER\_PATTERN/PROGRESSIVE\_DISCLOSURE]

- Navigation Behavior: [HOVER\_STATES/ACTIVE\_STATES/MEGA\_MENUS/STICKY\_NAVIGATION]

\*\*Search & Discovery:\*\*

- Search Functionality: [GLOBAL\_SEARCH/SCOPED\_SEARCH/FEDERATED\_SEARCH]

- Search Features: [AUTO\_COMPLETE/SUGGESTED\_SEARCHES/TYPO\_TOLERANCE/SYNONYMS]

- Filtering System: [FACETED\_SEARCH/DYNAMIC\_FILTERS/SAVED\_SEARCHES]

- Browse Mechanisms: [CATEGORY\_BROWSING/TAG\_BROWSING/RELATED\_CONTENT]

- Content Discovery: [FEATURED\_CONTENT/RECOMMENDATIONS/TRENDING/RECENT]

\*\*User Mental Models:\*\*

- User Research Integration: [CARD\_SORTING\_RESULTS/TREE\_TESTING\_INSIGHTS/USER\_INTERVIEWS]

- Task-Based Organization: [USER\_GOALS/TASK\_FLOWS/SCENARIO\_BASED\_STRUCTURE]

- Audience Segmentation: [USER\_TYPES/SKILL\_LEVELS/CONTEXT\_VARIATIONS]

- Language & Terminology: [USER\_VOCABULARY/INDUSTRY\_TERMS/PLAIN\_LANGUAGE]

- Conceptual Models: [HOW\_USERS\_THINK/EXPECTED\_RELATIONSHIPS/LOGICAL\_GROUPINGS]

\*\*Responsive Information Architecture:\*\*

- Mobile-First Strategy: [CONTENT\_PRIORITIZATION/PROGRESSIVE\_DISCLOSURE/SIMPLIFIED\_HIERARCHY]

- Cross-Device Consistency: [NAVIGATION\_ADAPTATION/CONTENT\_ADAPTATION/INTERACTION\_PATTERNS]

- Context-Aware Design: [LOCATION\_BASED/TIME\_BASED/DEVICE\_SPECIFIC\_CONTENT]

- Adaptive Navigation: [SCREEN\_SIZE\_OPTIMIZATION/TOUCH\_OPTIMIZATION/PERFORMANCE\_CONSIDERATIONS]

\*\*SEO & Findability:\*\*

- URL Strategy: [DESCRIPTIVE\_URLS/CANONICAL\_URLS/REDIRECT\_MANAGEMENT]

- Metadata Framework: [TITLE\_TAGS/META\_DESCRIPTIONS/STRUCTURED\_DATA]

- Internal Linking: [LINK\_HIERARCHY/ANCHOR\_TEXT/LINK\_EQUITY\_DISTRIBUTION]

- Site Structure: [XML\_SITEMAPS/ROBOTS\_TXT/CRAWLABILITY]

- Content Optimization: [KEYWORD\_INTEGRATION/CONTENT\_CLUSTERING/TOPIC\_AUTHORITY]

\*\*Content Management Integration:\*\*

- CMS Structure: [CONTENT\_TYPES/CUSTOM\_FIELDS/WORKFLOW\_INTEGRATION]

- Editorial Workflow: [CONTENT\_CREATION/REVIEW\_PROCESS/PUBLISHING\_WORKFLOW]

- Content Maintenance: [UPDATE\_SCHEDULES/CONTENT\_AUDIT/LINK\_MAINTENANCE]

- Multi-Author Management: [PERMISSIONS/ROLES/COLLABORATION\_TOOLS]

- Version Control: [CONTENT\_VERSIONING/ROLLBACK\_CAPABILITIES/CHANGE\_TRACKING]

\*\*Performance & Scalability:\*\*

- Information Scalability: [GROWTH\_ACCOMMODATION/STRUCTURAL\_FLEXIBILITY/MAINTENANCE\_EFFICIENCY]

- Navigation Performance: [MENU\_LOADING/SEARCH\_PERFORMANCE/FILTER\_EFFICIENCY]

- Content Loading: [LAZY\_LOADING/PROGRESSIVE\_ENHANCEMENT/CACHING\_STRATEGY]

- Search Performance: [INDEX\_OPTIMIZATION/QUERY\_EFFICIENCY/RESULT\_CACHING]

Provide complete information architecture with site maps, navigation specifications, content organization principles, and implementation guidelines for lovable.dev.

```

---

## Frontend Development Templates

### 10. Responsive Web Design Implementation

```

Act as a senior frontend developer specializing in responsive design for lovable.dev. Create a responsive implementation for [COMPONENT/PAGE] that:

\*\*Responsive Foundation:\*\*

- Layout Strategy: [MOBILE\_FIRST/DESKTOP\_FIRST/ADAPTIVE/HYBRID]

- Breakpoint System: [MOBILE\_320PX/TABLET\_768PX/DESKTOP\_1024PX/LARGE\_1440PX/CUSTOM]

- CSS Methodology: [BEM/ATOMIC\_CSS/CSS\_MODULES/STYLED\_COMPONENTS/TAILWIND]

- Grid System: [CSS\_GRID/FLEXBOX/HYBRID\_APPROACH/CUSTOM\_GRID]

- Typography Scale: [RESPONSIVE\_TYPOGRAPHY/FLUID\_TYPOGRAPHY/MODULAR\_SCALE]

\*\*Layout Implementation:\*\*

- Container Strategy: [FLUID\_CONTAINERS/FIXED\_CONTAINERS/HYBRID\_CONTAINERS]

- Component Layout: [CARD\_LAYOUTS/LIST\_LAYOUTS/GRID\_LAYOUTS/MASONRY\_LAYOUTS]

- Content Reflow: [CONTENT\_STACKING/CONTENT\_REARRANGEMENT/PROGRESSIVE\_DISCLOSURE]

- Sidebar Handling: [COLLAPSIBLE\_SIDEBAR/OFF\_CANVAS/DRAWER\_PATTERN/STACK\_CONVERSION]

- Header/Footer: [STICKY\_HEADER/COLLAPSIBLE\_HEADER/ADAPTIVE\_FOOTER]

\*\*Responsive Images & Media:\*\*

- Image Strategy: [SRCSET\_IMPLEMENTATION/PICTURE\_ELEMENT/ART\_DIRECTION]

- Image Optimization: [WEBP\_SUPPORT/LAZY\_LOADING/PROGRESSIVE\_ENHANCEMENT]

- Video Responsiveness: [RESPONSIVE\_EMBEDS/ASPECT\_RATIO\_MAINTENANCE/ADAPTIVE\_QUALITY]

- Icon Systems: [SVG\_ICONS/ICON\_FONTS/RESPONSIVE\_SIZING]

- Asset Loading: [CRITICAL\_ASSETS/NON\_CRITICAL\_LOADING/RESOURCE\_HINTS]

\*\*Interaction & Navigation:\*\*

- Touch Interactions: [TOUCH\_TARGETS/GESTURE\_SUPPORT/HOVER\_ALTERNATIVES]

- Navigation Patterns: [HAMBURGER\_MENU/BOTTOM\_NAV/TAB\_BAR/MEGA\_MENU]

- Form Optimization: [INPUT\_SIZING/KEYBOARD\_OPTIMIZATION/LABEL\_POSITIONING]

- Button Adaptation: [SIZE\_SCALING/SPACING\_ADJUSTMENT/TOUCH\_OPTIMIZATION]

- Scroll Behavior: [SMOOTH\_SCROLLING/SCROLL\_SNAPPING/INFINITE\_SCROLL]

\*\*Performance Optimization:\*\*

- Critical CSS: [ABOVE\_FOLD\_STYLES/INLINE\_CRITICAL\_CSS/DEFERRED\_LOADING]

- JavaScript Loading: [PROGRESSIVE\_ENHANCEMENT/LAZY\_LOADING/CODE\_SPLITTING]

- Resource Loading: [PRELOADING/PREFETCHING/RESOURCE\_PRIORITIZATION]

- Rendering Performance: [LAYOUT\_THRASHING/REFLOW\_OPTIMIZATION/PAINT\_OPTIMIZATION]

- Bundle Optimization: [TREE\_SHAKING/DEAD\_CODE\_ELIMINATION/COMPRESSION]

\*\*CSS Implementation:\*\*

```css

/\* Mobile-First Base Styles \*/

.component {

display: flex;

flex-direction: column;

gap: 1rem;

padding: 1rem;

margin: 0 auto;

max-width: 100%;

}

/\* Tablet Styles \*/

@media (min-width: 768px) {

.component {

flex-direction: row;

gap: 2rem;

padding: 2rem;

max-width: 768px;

}

}

/\* Desktop Styles \*/

@media (min-width: 1024px) {

.component {

gap: 3rem;

padding: 3rem;

max-width: 1200px;

}

}

/\* Large Desktop Styles \*/

@media (min-width: 1440px) {

.component {

gap: 4rem;

padding: 4rem;

max-width: 1440px;

}

}

```

\*\*Accessibility Integration:\*\*

- Screen Reader Support: [SEMANTIC\_HTML/ARIA\_LABELS/LANDMARK\_ROLES]

- Keyboard Navigation: [FOCUS\_MANAGEMENT/TAB\_ORDER/KEYBOARD\_SHORTCUTS]

- Color Contrast: [WCAG\_COMPLIANCE/HIGH\_CONTRAST\_MODE/COLOR\_ALTERNATIVES]

- Motion Preferences: [REDUCED\_MOTION/ANIMATION\_CONTROLS/STATIC\_ALTERNATIVES]

- Text Scaling: [ZOOM\_SUPPORT/TEXT\_RESIZE/LAYOUT\_ADAPTATION]

\*\*Testing Strategy:\*\*

- Device Testing: [REAL\_DEVICE\_TESTING/BROWSER\_TESTING/EMULATOR\_TESTING]

- Responsive Testing: [BREAKPOINT\_TESTING/CONTENT\_TESTING/INTERACTION\_TESTING]

- Performance Testing: [LIGHTHOUSE\_AUDITS/WEBPAGETEST/REAL\_USER\_MONITORING]

- Accessibility Testing: [SCREEN\_READER\_TESTING/KEYBOARD\_TESTING/CONTRAST\_TESTING]

- Cross-Browser Testing: [MODERN\_BROWSERS/LEGACY\_SUPPORT/PROGRESSIVE\_ENHANCEMENT]

\*\*AWS Amplify Integration:\*\*

- Build Configuration: [RESPONSIVE\_ASSET\_OPTIMIZATION/ENVIRONMENT\_CONFIGS]

- CDN Optimization: [IMAGE\_TRANSFORMATION/RESPONSIVE\_DELIVERY/CACHE\_OPTIMIZATION]

- Performance Monitoring: [CORE\_WEB\_VITALS/REAL\_USER\_MONITORING/PERFORMANCE\_BUDGETS]

- Deployment Strategy: [PREVIEW\_DEPLOYMENTS/FEATURE\_BRANCHES/ROLLBACK\_PROCEDURES]

Provide complete responsive implementation with HTML structure, CSS styles, JavaScript enhancements, and AWS Amplify deployment configuration.

```

### 11. Performance Optimization Implementation

```

Act as a frontend performance expert for lovable.dev implementations. Create a comprehensive performance optimization strategy for [APPLICATION/WEBSITE] that:

\*\*Performance Audit & Baseline:\*\*

- Current Performance Analysis: [LIGHTHOUSE\_AUDIT/WEBPAGETEST\_RESULTS/REAL\_USER\_METRICS]

- Core Web Vitals: [LCP\_MEASUREMENT/FID\_MEASUREMENT/CLS\_MEASUREMENT/INP\_TRACKING]

- Performance Budget: [RESOURCE\_LIMITS/TIMING\_BUDGETS/QUALITY\_THRESHOLDS]

- Bottleneck Identification: [CRITICAL\_PATH\_ANALYSIS/RESOURCE\_BLOCKING/RENDER\_BLOCKING]

- User Experience Impact: [BOUNCE\_RATE\_CORRELATION/CONVERSION\_IMPACT/USER\_SATISFACTION]

\*\*Loading Performance:\*\*

- Critical Resource Optimization: [CRITICAL\_CSS\_INLINING/CRITICAL\_JS\_PRIORITIZATION]

- Resource Loading Strategy: [PRELOAD/PREFETCH/MODULE\_PRELOAD/DNS\_PREFETCH]

- Asset Optimization: [IMAGE\_COMPRESSION/FONT\_OPTIMIZATION/MINIFICATION]

- Bundle Optimization: [CODE\_SPLITTING/TREE\_SHAKING/DYNAMIC\_IMPORTS]

- Caching Strategy: [HTTP\_CACHING/SERVICE\_WORKER\_CACHING/CDN\_CACHING]

\*\*Image & Media Optimization:\*\*

- Image Formats: [WEBP/AVIF/JPEG\_XL/FALLBACK\_FORMATS]

- Responsive Images: [SRCSET/PICTURE\_ELEMENT/ART\_DIRECTION]

- Lazy Loading: [INTERSECTION\_OBSERVER/NATIVE\_LAZY\_LOADING/PROGRESSIVE\_LOADING]

- Image Compression: [LOSSLESS\_COMPRESSION/LOSSY\_OPTIMIZATION/QUALITY\_ADJUSTMENT]

- Video Optimization: [STREAMING\_OPTIMIZATION/ADAPTIVE\_BITRATE/PRELOAD\_STRATEGIES]

\*\*JavaScript Performance:\*\*

- Code Splitting: [ROUTE\_BASED\_SPLITTING/COMPONENT\_BASED\_SPLITTING/VENDOR\_SPLITTING]

- Lazy Loading: [DYNAMIC\_IMPORTS/COMPONENT\_LAZY\_LOADING/ROUTE\_LAZY\_LOADING]

- Bundle Analysis: [WEBPACK\_BUNDLE\_ANALYZER/SOURCE\_MAP\_EXPLORER/BUNDLE\_SIZE\_TRACKING]

- Third-party Scripts: [THIRD\_PARTY\_OPTIMIZATION/ASYNC\_LOADING/DEFER\_LOADING]

- Runtime Performance: [EFFICIENT\_ALGORITHMS/MEMORY\_MANAGEMENT/EVENT\_OPTIMIZATION]

\*\*CSS Performance:\*\*

- CSS Delivery: [CRITICAL\_CSS/NON\_CRITICAL\_CSS\_DEFER/FONT\_DISPLAY\_OPTIMIZATION]

- CSS Optimization: [UNUSED\_CSS\_REMOVAL/CSS\_MINIFICATION/COMPRESSION]

- Animation Performance: [GPU\_ACCELERATION/TRANSFORM\_OPTIMIZATION/WILL\_CHANGE]

- Layout Performance: [LAYOUT\_THRASHING\_PREVENTION/REFLOW\_OPTIMIZATION]

- Font Loading: [FONT\_DISPLAY\_SWAP/FONT\_PRELOADING/FALLBACK\_FONTS]

\*\*Rendering Performance:\*\*

- First Paint Optimization: [CRITICAL\_RENDERING\_PATH/RENDER\_BLOCKING\_ELIMINATION]

- Layout Stability: [CLS\_PREVENTION/ASPECT\_RATIO\_BOXES/SIZE\_ATTRIBUTES]

- Paint Optimization: [COMPOSITE\_LAYERS/PAINT\_COMPLEXITY\_REDUCTION]

- Interaction Performance: [EVENT\_DELEGATION/DEBOUNCING/THROTTLING]

- Virtual Scrolling: [LARGE\_LIST\_OPTIMIZATION/WINDOWING/PAGINATION]

\*\*Network Performance:\*\*

- HTTP/2 Optimization: [MULTIPLEXING/SERVER\_PUSH/HEADER\_COMPRESSION]

- Connection Optimization: [KEEP\_ALIVE/CONNECTION\_POOLING/DOMAIN\_SHARDING\_AVOIDANCE]

- Compression: [GZIP/BROTLI/DYNAMIC\_COMPRESSION]

- CDN Strategy: [EDGE\_CACHING/GEOGRAPHIC\_DISTRIBUTION/CACHE\_INVALIDATION]

- Offline Strategy: [SERVICE\_WORKERS/CACHE\_API/OFFLINE\_FALLBACKS]

\*\*Monitoring & Measurement:\*\*

- Real User Monitoring: [RUM\_IMPLEMENTATION/PERFORMANCE\_ANALYTICS/USER\_EXPERIENCE\_TRACKING]

- Synthetic Monitoring: [LIGHTHOUSE\_CI/AUTOMATED\_TESTING/PERFORMANCE\_REGRESSION\_DETECTION]

- Performance Budgets: [BUDGET\_ENFORCEMENT/AUTOMATED\_ALERTS/CONTINUOUS\_MONITORING]

- Analytics Integration: [GOOGLE\_ANALYTICS/CUSTOM\_METRICS/BUSINESS\_IMPACT\_TRACKING]

- A/B Testing: [PERFORMANCE\_IMPACT\_TESTING/CONVERSION\_CORRELATION/OPTIMIZATION\_VALIDATION]

\*\*AWS Amplify Optimization:\*\*

- Build Optimization: [WEBPACK\_OPTIMIZATION/ROLLUP\_CONFIGURATION/VITE\_OPTIMIZATION]

- CDN Configuration: [CLOUDFRONT\_OPTIMIZATION/CACHE\_BEHAVIORS/COMPRESSION\_SETTINGS]

- Asset Hosting: [S3\_OPTIMIZATION/IMAGE\_TRANSFORMATION/RESPONSIVE\_IMAGES]

- Performance Monitoring: [CLOUDWATCH\_INTEGRATION/X\_RAY\_TRACING/CUSTOM\_METRICS]

- Environment Optimization: [PRODUCTION\_BUILDS/STAGING\_PERFORMANCE/DEVELOPMENT\_TOOLS]

\*\*Implementation Code Examples:\*\*

```javascript

// Critical Resource Loading

const loadCriticalCSS = () => {

const critical = document.createElement('style');

critical.textContent = `/\* Critical CSS \*/`;

document.head.appendChild(critical);

};

// Lazy Loading Images

const lazyImages = document.querySelectorAll('img[data-src]');

const imageObserver = new IntersectionObserver((entries) => {

entries.forEach(entry => {

if (entry.isIntersecting) {

const img = entry.target;

img.src = img.dataset.src;

img.removeAttribute('data-src');

imageObserver.unobserve(img);

}

});

});

lazyImages.forEach(img => imageObserver.observe(img));

// Service Worker for Caching

self.addEventListener('fetch', event => {

if (event.request.destination === 'image') {

event.respondWith(

caches.match(event.request).then(response => {

return response || fetch(event.request);

})

);

}

});

```

Provide complete performance optimization implementation with before/after metrics, monitoring setup, and continuous improvement strategies.

```

### 12. Component Library Development

```

Act as a component library architect for lovable.dev implementations. Create a comprehensive component library for [PROJECT/ORGANIZATION] that:

\*\*Library Architecture:\*\*

- Component Organization: [ATOMIC\_DESIGN/FEATURE\_BASED/DOMAIN\_DRIVEN/FLAT\_STRUCTURE]

- Build System: [ROLLUP/WEBPACK/VITE/MICROBUNDLE/CUSTOM\_BUILD]

- Package Structure: [MONOREPO/SINGLE\_PACKAGE/MULTIPLE\_PACKAGES]

- Distribution: [NPM\_PACKAGE/CDN\_DISTRIBUTION/DIRECT\_IMPORT/MULTIPLE\_FORMATS]

- Versioning Strategy: [SEMANTIC\_VERSIONING/INDEPENDENT\_VERSIONING/SYNCHRONIZED\_VERSIONING]

\*\*Design System Integration:\*\*

- Design Tokens: [COLOR\_TOKENS/TYPOGRAPHY\_TOKENS/SPACING\_TOKENS/ANIMATION\_TOKENS]

- Theme System: [DEFAULT\_THEME/DARK\_THEME/CUSTOM\_THEMES/RUNTIME\_THEMING]

- Branding: [BRAND\_VARIANTS/WHITE\_LABEL\_SUPPORT/CUSTOMIZATION\_OPTIONS]

- Consistency: [DESIGN\_GUIDELINES/USAGE\_PATTERNS/VISUAL\_CONSISTENCY]

- Accessibility: [WCAG\_COMPLIANCE/ARIA\_PATTERNS/KEYBOARD\_NAVIGATION/SCREEN\_READER\_SUPPORT]

\*\*Component Development:\*\*

- Base Components: [BUTTON/INPUT/SELECT/TEXTAREA/CHECKBOX/RADIO/TOGGLE]

- Layout Components: [CONTAINER/GRID/FLEX/STACK/SPACER/DIVIDER]

- Navigation Components: [NAVBAR/SIDEBAR/BREADCRUMB/PAGINATION/TABS/ACCORDION]

- Data Display: [TABLE/LIST/CARD/BADGE/TAG/TOOLTIP/POPOVER/MODAL]

- Feedback Components: [ALERT/TOAST/LOADING/PROGRESS/SKELETON/EMPTY\_STATE]

- Complex Components: [FORM/STEPPER/CAROUSEL/CALENDAR/CHART/DATA\_GRID]

\*\*Component API Design:\*\*

- Props Interface: [REQUIRED\_PROPS/OPTIONAL\_PROPS/DEFAULT\_VALUES/PROP\_VALIDATION]

- Event System: [CUSTOM\_EVENTS/EVENT\_NAMING/EVENT\_PAYLOAD/EVENT\_DELEGATION]

- Slot System: [NAMED\_SLOTS/DEFAULT\_SLOTS/SCOPED\_SLOTS/SLOT\_FALLBACKS]

- Composition: [COMPOUND\_COMPONENTS/RENDER\_PROPS/CHILDREN\_FUNCTIONS/FLEXIBLE\_API]

- Styling API: [CSS\_CLASSES/INLINE\_STYLES/CSS\_VARIABLES/STYLED\_PROPS]

\*\*Development Workflow:\*\*

- Component Creation: [COMPONENT\_GENERATOR/BOILERPLATE\_TEMPLATES/SCAFFOLDING\_TOOLS]

- Development Environment: [STORYBOOK/DOCUSAURUS/CUSTOM\_PLAYGROUND]

- Testing Strategy: [UNIT\_TESTS/VISUAL\_TESTS/ACCESSIBILITY\_TESTS/INTEGRATION\_TESTS]

- Quality Assurance: [LINTING/TYPE\_CHECKING/AUTOMATED\_TESTING/MANUAL\_REVIEW]

- Documentation: [COMPONENT\_DOCS/USAGE\_EXAMPLES/API\_REFERENCE/MIGRATION\_GUIDES]

\*\*Storybook Integration:\*\*

```javascript

// Component Story Example

export default {

title: 'Components/Button',

component: Button,

parameters: {

docs: {

description: {

component: 'Primary UI component for user interaction.'

}

}

},

argTypes: {

variant: {

control: { type: 'select' },

options: ['primary', 'secondary', 'tertiary']

},

size: {

control: { type: 'select' },

options: ['small', 'medium', 'large']

}

}

};

export const Primary = {

args: {

variant: 'primary',

children: 'Primary Button'

}

};

export const AllVariants = () => (

<div style={{ display: 'flex', gap: '1rem' }}>

<Button variant="primary">Primary</Button>

<Button variant="secondary">Secondary</Button>

<Button variant="tertiary">Tertiary</Button>

</div>

);

```

\*\*TypeScript Integration:\*\*

```typescript

// Component Interface

interface ButtonProps {

variant?: 'primary' | 'secondary' | 'tertiary';

size?: 'small' | 'medium' | 'large';

disabled?: boolean;

loading?: boolean;

children: React.ReactNode;

onClick?: (event: React.MouseEvent<HTMLButtonElement>) => void;

className?: string;

'data-testid'?: string;

}

// Component Implementation

export const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(

({ variant = 'primary', size = 'medium', disabled, loading, children, ...props }, ref) => {

const classes = clsx(

'btn',

`btn--${variant}`,

`btn--${size}`,

{ 'btn--loading': loading },

props.className

);

return (

<button

ref={ref}

className={classes}

disabled={disabled || loading}

{...props}

>

{loading ? <Spinner size="small" /> : children}

</button>

);

}

);

```

\*\*Testing Implementation:\*\*

```javascript

// Component Tests

import { render, fireEvent, screen } from '@testing-library/react';

import { Button } from './Button';

describe('Button', () => {

it('renders with correct text', () => {

render(<Button>Click me</Button>);

expect(screen.getByRole('button')).toHaveTextContent('Click me');

});

it('calls onClick when clicked', () => {

const handleClick = jest.fn();

render(<Button onClick={handleClick}>Click me</Button>);

fireEvent.click(screen.getByRole('button'));

expect(handleClick).toHaveBeenCalledTimes(1);

});

it('applies correct variant class', () => {

render(<Button variant="secondary">Click me</Button>);

expect(screen.getByRole('button')).toHaveClass('btn--secondary');

});

});

```

\*\*Build Configuration:\*\*

```javascript

// Rollup Configuration

export default {

input: 'src/index.ts',

output: [

{

file: 'dist/index.cjs.js',

format: 'cjs',

exports: 'named',

sourcemap: true

},

{

file: 'dist/index.esm.js',

format: 'esm',

exports: 'named',

sourcemap: true

}

],

plugins: [

typescript({ tsconfig: './tsconfig.json' }),

resolve({ extensions: ['.ts', '.tsx'] }),

commonjs(),

postcss({ extract: true, minimize: true })

],

external: ['react', 'react-dom']

};

```

\*\*Documentation & Distribution:\*\*

- API Documentation: [PROP\_TABLES/METHOD\_DOCUMENTATION/EVENT\_DOCUMENTATION]

- Usage Examples: [CODE\_EXAMPLES/LIVE\_DEMOS/COMMON\_PATTERNS/INTEGRATION\_GUIDES]

- Migration Guides: [VERSION\_UPGRADE\_GUIDES/BREAKING\_CHANGES/MIGRATION\_SCRIPTS]

- Contributing Guidelines: [DEVELOPMENT\_SETUP/CONTRIBUTION\_PROCESS/CODE\_STANDARDS]

- Release Process: [AUTOMATED\_RELEASES/CHANGELOG\_GENERATION/VERSION\_MANAGEMENT]

\*\*AWS Amplify Integration:\*\*

- Build Optimization: [COMPONENT\_TREE\_SHAKING/BUNDLE\_OPTIMIZATION/PEER\_DEPENDENCIES]

- Documentation Hosting: [STORYBOOK\_DEPLOYMENT/DOCS\_SITE\_HOSTING/CDN\_DISTRIBUTION]

- Package Distribution: [NPM\_PUBLISHING/CDN\_HOSTING/VERSION\_MANAGEMENT]

- CI/CD Integration: [AUTOMATED\_TESTING/VISUAL\_REGRESSION/AUTOMATED\_PUBLISHING]

Provide complete component library with build system, component examples, documentation structure, testing strategy, and distribution setup.

```

---

## Backend Development Templates

### 13. RESTful API Implementation

```

Act as a senior backend developer specializing in RESTful API design for windsurf. Create a comprehensive REST API for [PROJECT\_NAME] that:

\*\*API Architecture:\*\*

- API Design Pattern: [RESTFUL\_STANDARDS/HATEOAS/JSON\_API/CUSTOM\_CONVENTIONS]

- Resource Modeling: [RESOURCE\_IDENTIFICATION/HIERARCHY/RELATIONSHIPS/COLLECTIONS]

- HTTP Methods: [GET/POST/PUT/PATCH/DELETE/OPTIONS/HEAD]

- Status Codes: [SUCCESS\_CODES/CLIENT\_ERROR\_CODES/SERVER\_ERROR\_CODES]

- Content Negotiation: [JSON/XML/PROTOBUF/CUSTOM\_FORMATS]

\*\*URL Design & Routing:\*\*

- Resource URLs: [NOUN\_BASED\_URLS/HIERARCHICAL\_STRUCTURE/CLEAN\_URLS]

- Route Patterns: [COLLECTION\_ROUTES/RESOURCE\_ROUTES/NESTED\_ROUTES]

- Query Parameters: [FILTERING/SORTING/PAGINATION/SEARCH/FIELD\_SELECTION]

- Path Parameters: [RESOURCE\_IDS/NESTED\_RESOURCE\_ACCESS]

- URL Versioning: [VERSION\_IN\_URL/HEADER\_VERSIONING/CONTENT\_NEGOTIATION]

\*\*Request/Response Design:\*\*

- Request Structure: [HEADERS/BODY/PARAMETERS/VALIDATION]

- Response Format: [CONSISTENT\_STRUCTURE/ERROR\_FORMAT/SUCCESS\_FORMAT]

- Data Serialization: [JSON\_SERIALIZATION/XML\_SERIALIZATION/CUSTOM\_FORMATS]

- Pagination: [OFFSET\_PAGINATION/CURSOR\_PAGINATION/LINK\_HEADERS]

- Filtering & Searching: [QUERY\_SYNTAX/FIELD\_FILTERING/FULL\_TEXT\_SEARCH]

\*\*Authentication & Authorization:\*\*

- Authentication Methods: [JWT\_TOKENS/OAUTH2/API\_KEYS/BASIC\_AUTH]

- Authorization Patterns: [RBAC/ABAC/RESOURCE\_BASED/CUSTOM\_PERMISSIONS]

- Token Management: [TOKEN\_GENERATION/REFRESH\_TOKENS/TOKEN\_EXPIRY]

- Security Headers: [CORS/CSP/SECURITY\_HEADERS/RATE\_LIMITING]

- API Security: [INPUT\_VALIDATION/OUTPUT\_SANITIZATION/SQL\_INJECTION\_PREVENTION]

\*\*Error Handling:\*\*

- Error Response Format: [CONSISTENT\_ERROR\_STRUCTURE/ERROR\_CODES/ERROR\_MESSAGES]

- Validation Errors: [FIELD\_VALIDATION/BUSINESS\_RULE\_VALIDATION/FORMAT\_VALIDATION]

- HTTP Status Codes: [APPROPRIATE\_STATUS\_CODES/CUSTOM\_ERROR\_CODES]

- Error Logging: [ERROR\_TRACKING/LOG\_LEVELS/CONTEXT\_INFORMATION]

- Error Recovery: [RETRY\_MECHANISMS/FALLBACK\_RESPONSES/GRACEFUL\_DEGRADATION]

\*\*Data Validation:\*\*

- Input Validation: [SCHEMA\_VALIDATION/TYPE\_CHECKING/RANGE\_VALIDATION]

- Business Rule Validation: [DOMAIN\_CONSTRAINTS/RELATIONSHIP\_VALIDATION]

- Sanitization: [INPUT\_SANITIZATION/OUTPUT\_ENCODING/XSS\_PREVENTION]

- Custom Validators: [BUSINESS\_LOGIC\_VALIDATION/CROSS\_FIELD\_VALIDATION]

- Validation Middleware: [REQUEST\_VALIDATION/RESPONSE\_VALIDATION/ERROR\_HANDLING]

\*\*Performance Optimization:\*\*

- Caching Strategy: [HTTP\_CACHING/APPLICATION\_CACHING/DATABASE\_CACHING]

- Database Optimization: [QUERY\_OPTIMIZATION/INDEX\_USAGE/N+1\_PREVENTION]

- Response Optimization: [COMPRESSION/MINIFICATION/PARTIAL\_RESPONSES]

- Connection Management: [CONNECTION\_POOLING/KEEP\_ALIVE/TIMEOUT\_HANDLING]

- Resource Management: [MEMORY\_MANAGEMENT/CPU\_OPTIMIZATION/I/O\_OPTIMIZATION]

\*\*API Documentation:\*\*

```yaml

# OpenAPI Specification Example

openapi: 3.0.0

info:

title: Product API

version: 1.0.0

description: RESTful API for product management

paths:

/products:

get:

summary: List products

parameters:

- name: page

in: query

schema:

type: integer

default: 1

- name: limit

in: query

schema:

type: integer

default: 20

- name: category

in: query

schema:

type: string

responses:

'200':

description: Successful response

content:

application/json:

schema:

type: object

properties:

data:

type: array

items:

$ref: '#/components/schemas/Product'

pagination:

$ref: '#/components/schemas/Pagination'

post:

summary: Create product

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/ProductInput'

responses:

'201':

description: Product created

content:

application/json:

schema:

$ref: '#/components/schemas/Product'

components:

schemas:

Product:

type: object

properties:

id:

type: string

name:

type: string

price:

type: number

category:

type: string

createdAt:

type: string

format: date-time

```

\*\*Implementation Example:\*\*

```javascript

// Express.js Implementation

const express = require('express');

const app = express();

// Middleware

app.use(express.json());

app.use(cors());

app.use(helmet());

app.use(rateLimit({ windowMs: 15 \* 60 \* 1000, max: 100 }));

// Product Routes

app.get('/api/v1/products', async (req, res) => {

try {

const { page = 1, limit = 20, category, search } = req.query;

const filters = {};

if (category) filters.category = category;

if (search) filters.name = { $regex: search, $options: 'i' };

const products = await Product.find(filters)

.limit(limit \* 1)

.skip((page - 1) \* limit)

.sort({ createdAt: -1 });

const total = await Product.countDocuments(filters);

res.json({

data: products,

pagination: {

page: parseInt(page),

limit: parseInt(limit),

total,

pages: Math.ceil(total / limit)

}

});

} catch (error) {

res.status(500).json({

error: 'Internal server error',

message: error.message

});

}

});

app.post('/api/v1/products', validateProduct, async (req, res) => {

try {

const product = new Product(req.body);

await product.save();

res.status(201).json({

data: product,

message: 'Product created successfully'

});

} catch (error) {

if (error.name === 'ValidationError') {

res.status(400).json({

error: 'Validation error',

details: error.errors

});

} else {

res.status(500).json({

error: 'Internal server error',

message: error.message

});

}

}

});

```

\*\*Testing Strategy:\*\*

- Unit Tests: [CONTROLLER\_TESTS/SERVICE\_TESTS/UTILITY\_TESTS]

- Integration Tests: [API\_ENDPOINT\_TESTS/DATABASE\_INTEGRATION/THIRD\_PARTY\_INTEGRATION]

- Contract Tests: [API\_CONTRACT\_VALIDATION/CONSUMER\_DRIVEN\_CONTRACTS]

- Performance Tests: [LOAD\_TESTING/STRESS\_TESTING/BENCHMARKING]

- Security Tests: [PENETRATION\_TESTING/VULNERABILITY\_SCANNING/AUTH\_TESTING]

\*\*AWS Integration:\*\*

- Lambda Deployment: [SERVERLESS\_DEPLOYMENT/COLD\_START\_OPTIMIZATION]

- API Gateway: [API\_GATEWAY\_INTEGRATION/CUSTOM\_AUTHORIZERS/THROTTLING]

- Database Integration: [RDS\_CONNECTION/DYNAMODB\_INTEGRATION/CONNECTION\_POOLING]

- Monitoring: [CLOUDWATCH\_METRICS/X\_RAY\_TRACING/CUSTOM\_METRICS]

- Cost Optimization: [REQUEST\_EFFICIENCY/RESOURCE\_OPTIMIZATION/USAGE\_MONITORING]

Provide complete REST API implementation with routing, validation, error handling, documentation, testing, and AWS deployment configuration.

```

### 14. Database Operations & Optimization

```

Act as a database architect and optimization expert for windsurf. Create a comprehensive database strategy for [PROJECT\_NAME] that:

\*\*Database Architecture:\*\*

- Database Selection: [POSTGRESQL/MYSQL/MONGODB/DYNAMODB/HYBRID\_APPROACH]

- Architecture Pattern: [SINGLE\_DATABASE/DATABASE\_PER\_SERVICE/SHARED\_DATABASE]

- Deployment Strategy: [CLOUD\_MANAGED/SELF\_HOSTED/HYBRID/MULTI\_REGION]

- Backup Strategy: [AUTOMATED\_BACKUPS/POINT\_IN\_TIME\_RECOVERY/CROSS\_REGION\_BACKUP]

- Disaster Recovery: [RTO\_REQUIREMENTS/RPO\_REQUIREMENTS/FAILOVER\_PROCEDURES]

\*\*Schema Design & Optimization:\*\*

- Entity Modeling: [ENTITY\_RELATIONSHIPS/NORMALIZATION/DENORMALIZATION\_STRATEGIES]

- Table Design: [COLUMN\_DEFINITIONS/DATA\_TYPES/CONSTRAINTS/INDEXES]

- Relationship Management: [FOREIGN\_KEYS/JUNCTION\_TABLES/REFERENTIAL\_INTEGRITY]

- Schema Evolution: [MIGRATION\_STRATEGIES/BACKWARD\_COMPATIBILITY/VERSION\_CONTROL]

- Data Modeling: [CONCEPTUAL\_MODEL/LOGICAL\_MODEL/PHYSICAL\_MODEL]

\*\*Query Optimization:\*\*

- Query Performance: [EXECUTION\_PLANS/INDEX\_OPTIMIZATION/QUERY\_REWRITING]

- Index Strategy: [PRIMARY\_INDEXES/SECONDARY\_INDEXES/COMPOSITE\_INDEXES/PARTIAL\_INDEXES]

- Query Patterns: [COMMON\_QUERIES/OPTIMIZATION\_TECHNIQUES/QUERY\_CACHING]

- Database Statistics: [STATISTICS\_MAINTENANCE/QUERY\_ANALYSIS/PERFORMANCE\_MONITORING]

- Slow Query Analysis: [QUERY\_LOGGING/PERFORMANCE\_PROFILING/BOTTLENECK\_IDENTIFICATION]

\*\*Performance Tuning:\*\*

- Connection Management: [CONNECTION\_POOLING/CONNECTION\_LIMITS/IDLE\_TIMEOUTS]

- Memory Optimization: [BUFFER\_POOL\_TUNING/CACHE\_OPTIMIZATION/MEMORY\_ALLOCATION]

- Storage Optimization: [DISK\_I/O\_OPTIMIZATION/STORAGE\_ENGINE\_SELECTION/COMPRESSION]

- Configuration Tuning: [DATABASE\_PARAMETERS/PERFORMANCE\_SETTINGS/RESOURCE\_ALLOCATION]

- Monitoring & Alerting: [PERFORMANCE\_METRICS/HEALTH\_MONITORING/AUTOMATED\_ALERTS]

\*\*Data Access Patterns:\*\*

- ORM Integration: [ORM\_SELECTION/QUERY\_OPTIMIZATION/N+1\_PREVENTION]

- Repository Pattern: [DATA\_ACCESS\_ABSTRACTION/BUSINESS\_LOGIC\_SEPARATION]

- Caching Strategy: [QUERY\_CACHING/RESULT\_CACHING/DISTRIBUTED\_CACHING]

- Connection Pooling: [POOL\_SIZE\_OPTIMIZATION/CONNECTION\_LIFECYCLE/HEALTH\_CHECKS]

- Transaction Management: [ACID\_PROPERTIES/ISOLATION\_LEVELS/DEADLOCK\_PREVENTION]

\*\*Scalability Solutions:\*\*

- Read Scaling: [READ\_REPLICAS/LOAD\_BALANCING/QUERY\_ROUTING]

- Write Scaling: [WRITE\_SHARDING/PARTITIONING/FEDERATION]

- Horizontal Scaling: [SHARDING\_STRATEGIES/CONSISTENT\_HASHING/DISTRIBUTED\_QUERIES]

- Vertical Scaling: [RESOURCE\_SCALING/PERFORMANCE\_TUNING/CAPACITY\_PLANNING]

- Caching Layers: [APPLICATION\_CACHE/DATABASE\_CACHE/DISTRIBUTED\_CACHE]

\*\*Implementation Examples:\*\*

```sql

-- Database Schema with Optimization

CREATE TABLE users (

id SERIAL PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

username VARCHAR(50) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

is\_active BOOLEAN DEFAULT true

);

-- Optimized Indexes

CREATE INDEX idx\_users\_email ON users(email) WHERE is\_active = true;

CREATE INDEX idx\_users\_username ON users(username) WHERE is\_active = true;

CREATE INDEX idx\_users\_created\_at ON users(created\_at DESC);

-- Partitioned Table for Large Data

CREATE TABLE user\_activities (

id BIGSERIAL,

user\_id INTEGER REFERENCES users(id),

activity\_type VARCHAR(50) NOT NULL,

activity\_data JSONB,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (id, created\_at)

) PARTITION BY RANGE (created\_at);

-- Monthly Partitions

CREATE TABLE user\_activities\_2024\_01 PARTITION OF user\_activities

FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

-- Optimized Query Examples

-- Using prepared statements and proper indexing

EXPLAIN ANALYZE

SELECT u.id, u.username, COUNT(ua.id) as activity\_count

FROM users u

LEFT JOIN user\_activities ua ON u.id = ua.user\_id

WHERE u.is\_active = true

AND u.created\_at >= '2024-01-01'

GROUP BY u.id, u.username

ORDER BY activity\_count DESC

LIMIT 20;

```

```javascript

// Database Connection and Optimization

const { Pool } = require('pg');

// Optimized Connection Pool

const pool = new Pool({

host: process.env.DB\_HOST,

port: process.env.DB\_PORT,

database: process.env.DB\_NAME,

user: process.env.DB\_USER,

password: process.env.DB\_PASSWORD,

max: 20, // Maximum pool size

idleTimeoutMillis: 30000, // 30 seconds

connectionTimeoutMillis: 2000, // 2 seconds

maxUses: 7500, // Rotate connections

});

// Repository Pattern with Optimization

class UserRepository {

async findById(id) {

const query = `

SELECT id, email, username, created\_at, is\_active

FROM users

WHERE id = $1 AND is\_active = true

`;

const result = await pool.query(query, [id]);

return result.rows[0];

}

async findWithActivities(userId, limit = 10) {

const query = `

SELECT

u.id, u.username,

json\_agg(

json\_build\_object(

'id', ua.id,

'type', ua.activity\_type,

'data', ua.activity\_data,

'created\_at', ua.created\_at

) ORDER BY ua.created\_at DESC

) FILTER (WHERE ua.id IS NOT NULL) as activities

FROM users u

LEFT JOIN (

SELECT \* FROM user\_activities

WHERE user\_id = $1

ORDER BY created\_at DESC

LIMIT $2

) ua ON u.id = ua.user\_id

WHERE u.id = $1 AND u.is\_active = true

GROUP BY u.id, u.username

`;

const result = await pool.query(query, [userId, limit]);

return result.rows[0];

}

async createUser(userData) {

const client = await pool.connect();

try {

await client.query('BEGIN');

const insertQuery = `

INSERT INTO users (email, username, password\_hash)

VALUES ($1, $2, $3)

RETURNING id, email, username, created\_at

`;

const result = await client.query(insertQuery, [

userData.email,

userData.username,

userData.passwordHash

]);

await client.query('COMMIT');

return result.rows[0];

} catch (error) {

await client.query('ROLLBACK');

throw error;

} finally {

client.release();

}

}

async batchInsertActivities(activities) {

const query = `

INSERT INTO user\_activities (user\_id, activity\_type, activity\_data, created\_at)

VALUES ${activities.map((\_, i) => `($${i \* 4 + 1}, $${i \* 4 + 2}, $${i \* 4 + 3}, $${i \* 4 + 4})`).join(', ')}

ON CONFLICT DO NOTHING

RETURNING id

`;

const values = activities.flatMap(activity => [

activity.userId,

activity.type,

JSON.stringify(activity.data),

activity.createdAt

]);

const result = await pool.query(query, values);

return result.rows;

}

}

```

\*\*Caching Implementation:\*\*

```javascript

const Redis = require('redis');

class CacheService {

constructor() {

this.redis = Redis.createClient({

host: process.env.REDIS\_HOST,

port: process.env.REDIS\_PORT,

retryDelayOnFailover: 100,

maxRetriesPerRequest: 3

});

}

async getOrSet(key, fetchFunction, ttl = 3600) {

try {

// Try to get from cache

const cached = await this.redis.get(key);

if (cached) {

return JSON.parse(cached);

}

// Fetch from database

const data = await fetchFunction();

// Cache the result

await this.redis.setex(key, ttl, JSON.stringify(data));

return data;

} catch (error) {

console.error('Cache error:', error);

// Fallback to direct database call

return await fetchFunction();

}

}

async invalidatePattern(pattern) {

const keys = await this.redis.keys(pattern);

if (keys.length > 0) {

await this.redis.del(keys);

}

}

}

```

\*\*Migration Management:\*\*

```javascript

// Database Migration System

class MigrationRunner {

async runMigrations() {

const migrations = await this.getPendingMigrations();

for (const migration of migrations) {

console.log(`Running migration: ${migration.name}`);

const client = await pool.connect();

try {

await client.query('BEGIN');

await migration.up(client);

await this.recordMigration(client, migration.name);

await client.query('COMMIT');

console.log(`Completed migration: ${migration.name}`);

} catch (error) {

await client.query('ROLLBACK');

console.error(`Migration failed: ${migration.name}`, error);

throw error;

} finally {

client.release();

}

}

}

}

```

\*\*AWS RDS Integration:\*\*

- RDS Configuration: [INSTANCE\_SIZING/STORAGE\_OPTIMIZATION/BACKUP\_CONFIGURATION]

- Performance Insights: [QUERY\_MONITORING/PERFORMANCE\_ANALYSIS/BOTTLENECK\_IDENTIFICATION]

- Parameter Groups: [CUSTOM\_PARAMETERS/PERFORMANCE\_TUNING/OPTIMIZATION\_SETTINGS]

- Security: [VPC\_SECURITY/ENCRYPTION/ACCESS\_CONTROL/AUDIT\_LOGGING]

- Monitoring: [CLOUDWATCH\_METRICS/CUSTOM\_DASHBOARDS/AUTOMATED\_ALERTS]

\*\*Cost Optimization:\*\*

- Resource Right-sizing: [INSTANCE\_OPTIMIZATION/STORAGE\_OPTIMIZATION/COMPUTE\_EFFICIENCY]

- Query Optimization: [EXPENSIVE\_QUERY\_IDENTIFICATION/INDEX\_OPTIMIZATION/EXECUTION\_PLANNING]

- Connection Efficiency: [POOLING\_OPTIMIZATION/IDLE\_CONNECTION\_MANAGEMENT/RESOURCE\_REUSE]

- Monitoring Costs: [USAGE\_TRACKING/COST\_ALERTS/OPTIMIZATION\_RECOMMENDATIONS]

Provide complete database implementation with schema design, optimization strategies, caching, migration management, and AWS integration.

```

### 15. Authentication & Authorization Implementation

```

Act as a security architect specializing in authentication and authorization for windsurf. Create a comprehensive auth system for [PROJECT\_NAME] that:

\*\*Authentication Architecture:\*\*

- Authentication Methods: [JWT/OAUTH2/SAML/MULTI\_FACTOR/PASSWORDLESS]

- Token Strategy: [ACCESS\_TOKENS/REFRESH\_TOKENS/SESSION\_TOKENS/API\_KEYS]

- Identity Providers: [INTERNAL\_AUTH/SOCIAL\_LOGIN/ENTERPRISE\_SSO/FEDERATED\_IDENTITY]

- Session Management: [STATELESS\_JWT/STATEFUL\_SESSIONS/HYBRID\_APPROACH]

- Multi-Factor Authentication: [TOTP/SMS/EMAIL/BIOMETRIC/HARDWARE\_TOKENS]

\*\*Authorization Framework:\*\*

- Authorization Model: [RBAC/ABAC/ACL/CUSTOM\_PERMISSIONS]

- Permission System: [RESOURCE\_BASED/ACTION\_BASED/HIERARCHICAL\_PERMISSIONS]

- Role Management: [STATIC\_ROLES/DYNAMIC\_ROLES/ROLE\_INHERITANCE]

- Policy Engine: [RULE\_BASED\_POLICIES/ATTRIBUTE\_BASED\_POLICIES/CUSTOM\_LOGIC]

- Access Control: [API\_LEVEL/RESOURCE\_LEVEL/FIELD\_LEVEL/ROW\_LEVEL]

\*\*Security Implementation:\*\*

- Password Security: [HASHING\_ALGORITHMS/SALT\_GENERATION/PASSWORD\_POLICIES]

- Token Security: [JWT\_SIGNING/ENCRYPTION/TOKEN\_ROTATION/REVOCATION]

- Secure Storage: [CREDENTIAL\_STORAGE/SECRET\_MANAGEMENT/KEY\_ROTATION]

- Transport Security: [HTTPS\_ONLY/HSTS/CERTIFICATE\_PINNING]

- Attack Prevention: [BRUTE\_FORCE\_PROTECTION/RATE\_LIMITING/CSRF\_PROTECTION]

\*\*Implementation Examples:\*\*

```javascript

// JWT Authentication Service

const jwt = require('jsonwebtoken');

const bcrypt = require('bcrypt');

const crypto = require('crypto');

class AuthService {

constructor() {

this.jwtSecret = process.env.JWT\_SECRET;

this.jwtRefreshSecret = process.env.JWT\_REFRESH\_SECRET;

this.accessTokenExpiry = '15m';

this.refreshTokenExpiry = '7d';

}

async hashPassword(password) {

const saltRounds = 12;

return await bcrypt.hash(password, saltRounds);

}

async verifyPassword(password, hashedPassword) {

return await bcrypt.compare(password, hashedPassword);

}

generateTokens(payload) {

const accessToken = jwt.sign(

payload,

this.jwtSecret,

{

expiresIn: this.accessTokenExpiry,

issuer: 'windsurf-api',

audience: 'windsurf-client'

}

);

const refreshToken = jwt.sign(

{ userId: payload.userId, tokenId: crypto.randomUUID() },

this.jwtRefreshSecret,

{

expiresIn: this.refreshTokenExpiry,

issuer: 'windsurf-api'

}

);

return { accessToken, refreshToken };

}

verifyAccessToken(token) {

try {

return jwt.verify(token, this.jwtSecret, {

issuer: 'windsurf-api',

audience: 'windsurf-client'

});

} catch (error) {

throw new Error('Invalid access token');

}

}

verifyRefreshToken(token) {

try {

return jwt.verify(token, this.jwtRefreshSecret, {

issuer: 'windsurf-api'

});

} catch (error) {

throw new Error('Invalid refresh token');

}

}

async login(email, password) {

// Find user

const user = await User.findOne({ email, isActive: true });

if (!user) {

throw new Error('Invalid credentials');

}

// Verify password

const isValidPassword = await this.verifyPassword(password, user.passwordHash);

if (!isValidPassword) {

throw new Error('Invalid credentials');

}

// Check for account lockout

if (user.lockedUntil && user.lockedUntil > Date.now()) {

throw new Error('Account temporarily locked');

}

// Reset failed attempts on successful login

if (user.failedLoginAttempts > 0) {

await User.updateOne(

{ \_id: user.\_id },

{ $unset: { failedLoginAttempts: 1, lockedUntil: 1 } }

);

}

// Generate tokens

const tokenPayload = {

userId: user.\_id,

email: user.email,

roles: user.roles,

permissions: user.permissions

};

const tokens = this.generateTokens(tokenPayload);

// Store refresh token

await RefreshToken.create({

userId: user.\_id,

token: tokens.refreshToken,

expiresAt: new Date(Date.now() + 7 \* 24 \* 60 \* 60 \* 1000) // 7 days

});

return {

user: {

id: user.\_id,

email: user.email,

roles: user.roles

},

...tokens

};

}

async refreshAccessToken(refreshToken) {

// Verify refresh token

const decoded = this.verifyRefreshToken(refreshToken);

// Check if refresh token exists and is valid

const storedToken = await RefreshToken.findOne({

userId: decoded.userId,

token: refreshToken,

expiresAt: { $gt: new Date() }

});

if (!storedToken) {

throw new Error('Invalid refresh token');

}

// Get user

const user = await User.findById(decoded.userId);

if (!user || !user.isActive) {

throw new Error('User not found or inactive');

}

// Generate new access token

const tokenPayload = {

userId: user.\_id,

email: user.email,

roles: user.roles,

permissions: user.permissions

};

const { accessToken } = this.generateTokens(tokenPayload);

return { accessToken };

}

async logout(refreshToken) {

if (refreshToken) {

await RefreshToken.deleteOne({ token: refreshToken });

}

}

}

```

```javascript

// Authorization Middleware

class AuthorizationService {

constructor() {

this.rolePermissions = new Map();

this.initializeRolePermissions();

}

initializeRolePermissions() {

// Define role-based permissions

this.rolePermissions.set('admin', [

'users:read', 'users:write', 'users:delete',

'products:read', 'products:write', 'products:delete',

'orders:read', 'orders:write', 'orders:delete'

]);

this.rolePermissions.set('manager', [

'users:read', 'products:read', 'products:write',

'orders:read', 'orders:write'

]);

this.rolePermissions.set('employee', [

'products:read', 'orders:read'

]);

this.rolePermissions.set('customer', [

'products:read', 'orders:read:own'

]);

}

hasPermission(userRoles, requiredPermission, context = {}) {

// Get all permissions for user roles

const userPermissions = userRoles.flatMap(role =>

this.rolePermissions.get(role) || []

);

// Check for exact permission match

if (userPermissions.includes(requiredPermission)) {

return true;

}

// Check for contextual permissions (e.g., own resources)

if (requiredPermission.includes(':own') && context.userId) {

const basePermission = requiredPermission.replace(':own', '');

return userPermissions.includes(basePermission) &&

context.resourceOwnerId === context.userId;

}

return false;

}

createAuthMiddleware(requiredPermission) {

return async (req, res, next) => {

try {

// Check if user is authenticated

if (!req.user) {

return res.status(401).json({ error: 'Authentication required' });

}

// Prepare context for authorization

const context = {

userId: req.user.userId,

resourceOwnerId: req.params.userId || req.body.userId

};

// Check permissions

const hasAccess = this.hasPermission(

req.user.roles,

requiredPermission,

context

);

if (!hasAccess) {

return res.status(403).json({

error: 'Insufficient permissions',

required: requiredPermission

});

}

next();

} catch (error) {

res.status(500).json({ error: 'Authorization error' });

}

};

}

}

// Authentication Middleware

const authenticateJWT = async (req, res, next) => {

try {

const authHeader = req.headers.authorization;

if (!authHeader || !authHeader.startsWith('Bearer ')) {

return res.status(401).json({ error: 'Access token required' });

}

const token = authHeader.substring(7);

const authService = new AuthService();

const decoded = authService.verifyAccessToken(token);

// Optionally verify user still exists and is active

const user = await User.findById(decoded.userId);

if (!user || !user.isActive) {

return res.status(401).json({ error: 'Invalid user' });

}

req.user = decoded;

next();

} catch (error) {

res.status(401).json({ error: 'Invalid access token' });

}

};

```

```javascript

// Rate Limiting and Brute Force Protection

const rateLimit = require('express-rate-limit');

const MongoStore = require('rate-limit-mongo');

// Rate limiting middleware

const createRateLimiter = (windowMs, max, message) => {

return rateLimit({

store: new MongoStore({

uri: process.env.MONGODB\_URI,

collectionName: 'rate\_limits',

expireTimeMs: windowMs

}),

windowMs,

max,

message: { error: message },

standardHeaders: true,

legacyHeaders: false

});

};

// Different rate limits for different endpoints

const authRateLimit = createRateLimiter(

15 \* 60 \* 1000, // 15 minutes

5, // 5 attempts

'Too many authentication attempts'

);

const apiRateLimit = createRateLimiter(

60 \* 1000, // 1 minute

100, // 100 requests

'Too many API requests'

);

// Brute force protection

class BruteForceProtection {

async recordFailedAttempt(userId) {

const user = await User.findById(userId);

if (!user) return;

const attempts = (user.failedLoginAttempts || 0) + 1;

const updates = { failedLoginAttempts: attempts };

// Lock account after 5 failed attempts for 30 minutes

if (attempts >= 5) {

updates.lockedUntil = Date.now() + 30 \* 60 \* 1000; // 30 minutes

}

await User.updateOne({ \_id: userId }, updates);

}

async isAccountLocked(userId) {

const user = await User.findById(userId);

return user && user.lockedUntil && user.lockedUntil > Date.now();

}

}

```

\*\*OAuth2 Implementation:\*\*

```javascript

// OAuth2 Service

class OAuth2Service {

constructor() {

this.providers = {

google: {

clientId: process.env.GOOGLE\_CLIENT\_ID,

clientSecret: process.env.GOOGLE\_CLIENT\_SECRET,

redirectUri: process.env.GOOGLE\_REDIRECT\_URI

},

github: {

clientId: process.env.GITHUB\_CLIENT\_ID,

clientSecret: process.env.GITHUB\_CLIENT\_SECRET,

redirectUri: process.env.GITHUB\_REDIRECT\_URI

}

};

}

getAuthorizationUrl(provider, state) {

const config = this.providers[provider];

if (!config) throw new Error('Unsupported provider');

const params = new URLSearchParams({

client\_id: config.clientId,

redirect\_uri: config.redirectUri,

state: state,

scope: this.getScope(provider),

response\_type: 'code'

});

return `${this.getAuthUrl(provider)}?${params.toString()}`;

}

async handleCallback(provider, code, state) {

// Verify state parameter

if (!this.verifyState(state)) {

throw new Error('Invalid state parameter');

}

// Exchange code for access token

const tokenData = await this.exchangeCodeForToken(provider, code);

// Get user profile

const profile = await this.getUserProfile(provider, tokenData.access\_token);

// Find or create user

let user = await User.findOne({

$or: [

{ email: profile.email },

{ [`oauth.${provider}.id`]: profile.id }

]

});

if (!user) {

user = await User.create({

email: profile.email,

username: profile.username || profile.email.split('@')[0],

profile: {

name: profile.name,

avatar: profile.avatar

},

oauth: {

[provider]: {

id: profile.id,

username: profile.username

}

},

isActive: true

});

}

// Generate JWT tokens

const authService = new AuthService();

return authService.generateTokens({

userId: user.\_id,

email: user.email,

roles: user.roles || ['customer']

});

}

}

```

\*\*Security Monitoring:\*\*

```javascript

// Security Event Logging

class SecurityLogger {

async logAuthEvent(event, userId, metadata = {}) {

await SecurityLog.create({

event,

userId,

timestamp: new Date(),

ip: metadata.ip,

userAgent: metadata.userAgent,

success: metadata.success,

details: metadata.details

});

// Check for suspicious activity

if (event === 'LOGIN\_FAILED') {

await this.checkSuspiciousActivity(userId, metadata.ip);

}

}

async checkSuspiciousActivity(userId, ip) {

const recentFailures = await SecurityLog.countDocuments({

event: 'LOGIN\_FAILED',

$or: [{ userId }, { ip }],

timestamp: {

$gte: new Date(Date.now() - 60 \* 60 \* 1000) // Last hour

}

});

if (recentFailures > 10) {

// Alert security team

await this.sendSecurityAlert({

type: 'SUSPICIOUS\_LOGIN\_ACTIVITY',

userId,

ip,

failureCount: recentFailures

});

}

}

}

```

\*\*AWS Integration:\*\*

- Cognito Integration: [USER\_POOLS/IDENTITY\_POOLS/FEDERATED\_IDENTITIES]

- IAM Integration: [ROLE\_BASED\_ACCESS/RESOURCE\_POLICIES/TEMPORARY\_CREDENTIALS]

- Secrets Manager: [SECRET\_ROTATION/SECURE\_STORAGE/ACCESS\_CONTROL]

- WAF Integration: [REQUEST\_FILTERING/RATE\_LIMITING/GEO\_BLOCKING]

- CloudTrail: [AUDIT\_LOGGING/SECURITY\_MONITORING/COMPLIANCE\_REPORTING]

Provide complete authentication and authorization system with JWT implementation, RBAC, security monitoring, and AWS integration.

```

---

## Full-Stack Integration Templates

### 16. Frontend-Backend API Integration

```

Act as a full-stack integration expert specializing in lovable.dev and windsurf. Create a comprehensive API integration for [PROJECT\_NAME] that:

\*\*Integration Architecture:\*\*

- API Communication: [REST/GRAPHQL/WEBSOCKETS/HYBRID\_APPROACH]

- Data Flow: [REQUEST\_RESPONSE/REAL\_TIME\_UPDATES/BATCH\_PROCESSING/EVENT\_DRIVEN]

- State Management: [CLIENT\_STATE/SERVER\_STATE/SYNCHRONIZED\_STATE/OPTIMISTIC\_UPDATES]

- Error Handling: [CENTRALIZED\_ERROR\_HANDLING/RETRY\_MECHANISMS/FALLBACK\_STRATEGIES]

- Authentication Flow: [TOKEN\_BASED/SESSION\_BASED/OAUTH\_FLOW/API\_KEY\_AUTHENTICATION]

\*\*Frontend API Client (Lovable):\*\*

```typescript

// API Client Configuration

import axios, { AxiosInstance, AxiosRequestConfig, AxiosResponse } from 'axios';

interface ApiResponse<T> {

data: T;

message?: string;

success: boolean;

}

interface PaginatedResponse<T> extends ApiResponse<T[]> {

pagination: {

page: number;

limit: number;

total: number;

pages: number;

};

}

class ApiClient {

private client: AxiosInstance;

private baseURL: string;

private authToken: string | null = null;

constructor(baseURL: string) {

this.baseURL = baseURL;

this.client = axios.create({

baseURL,

timeout: 10000,

headers: {

'Content-Type': 'application/json',

},

});

this.setupInterceptors();

}

private setupInterceptors() {

// Request interceptor

this.client.interceptors.request.use(

(config) => {

if (this.authToken) {

config.headers.Authorization = `Bearer ${this.authToken}`;

}

return config;

},

(error) => Promise.reject(error)

);

// Response interceptor

this.client.interceptors.response.use(

(response) => response,

async (error) => {

const originalRequest = error.config;

if (error.response?.status === 401 && !originalRequest.\_retry) {

originalRequest.\_retry = true;

try {

const newToken = await this.refreshToken();

this.setAuthToken(newToken);

originalRequest.headers.Authorization = `Bearer ${newToken}`;

return this.client(originalRequest);

} catch (refreshError) {

this.handleAuthError();

return Promise.reject(refreshError);

}

}

return Promise.reject(error);

}

);

}

setAuthToken(token: string) {

this.authToken = token;

localStorage.setItem('authToken', token);

}

private async refreshToken(): Promise<string> {

const refreshToken = localStorage.getItem('refreshToken');

if (!refreshToken) throw new Error('No refresh token');

const response = await axios.post(`${this.baseURL}/auth/refresh`, {

refreshToken,

});

return response.data.accessToken;

}

private handleAuthError() {

this.authToken = null;

localStorage.removeItem('authToken');

localStorage.removeItem('refreshToken');

window.location.href = '/login';

}

async get<T>(url: string, config?: AxiosRequestConfig): Promise<ApiResponse<T>> {

const response = await this.client.get(url, config);

return response.data;

}

async post<T>(

url: string,

data?: any,

config?: AxiosRequestConfig

): Promise<ApiResponse<T>> {

const response = await this.client.post(url, data, config);

return response.data;

}

async put<T>(

url: string,

data?: any,

config?: AxiosRequestConfig

): Promise<ApiResponse<T>> {

const response = await this.client.put(url, data, config);

return response.data;

}

async delete<T>(url: string, config?: AxiosRequestConfig): Promise<ApiResponse<T>> {

const response = await this.client.delete(url, config);

return response.data;

}

}

// API Service Classes

export class UserService {

constructor(private apiClient: ApiClient) {}

async getUsers(params?: {

page?: number;

limit?: number;

search?: string;

}): Promise<PaginatedResponse<User>> {

return this.apiClient.get('/users', { params });

}

async getUserById(id: string): Promise<ApiResponse<User>> {

return this.apiClient.get(`/users/${id}`);

}

async createUser(userData: CreateUserRequest): Promise<ApiResponse<User>> {

return this.apiClient.post('/users', userData);

}

async updateUser(id: string, userData: UpdateUserRequest): Promise<ApiResponse<User>> {

return this.apiClient.put(`/users/${id}`, userData);

}

async deleteUser(id: string): Promise<ApiResponse<void>> {

return this.apiClient.delete(`/users/${id}`);

}

}

```

\*\*React Integration with React Query:\*\*

```typescript

// React Query Integration

import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';

// Custom Hooks for API Integration

export function useUsers(params?: { page?: number; limit?: number; search?: string }) {

return useQuery({

queryKey: ['users', params],

queryFn: () => userService.getUsers(params),

staleTime: 5 \* 60 \* 1000, // 5 minutes

cacheTime: 10 \* 60 \* 1000, // 10 minutes

});

}

export function useUser(id: string) {

return useQuery({

queryKey: ['user', id],

queryFn: () => userService.getUserById(id),

enabled: !!id,

});

}

export function useCreateUser() {

const queryClient = useQueryClient();

return useMutation({

mutationFn: (userData: CreateUserRequest) => userService.createUser(userData),

onSuccess: () => {

queryClient.invalidateQueries({ queryKey: ['users'] });

},

onError: (error) => {

console.error('Failed to create user:', error);

},

});

}

export function useUpdateUser() {

const queryClient = useQueryClient();

return useMutation({

mutationFn: ({ id, userData }: { id: string; userData: UpdateUserRequest }) =>

userService.updateUser(id, userData),

onSuccess: (data, variables) => {

queryClient.invalidateQueries({ queryKey: ['users'] });

queryClient.invalidateQueries({ queryKey: ['user', variables.id] });

},

});

}

// React Component with API Integration

import React, { useState } from 'react';

import { useUsers, useCreateUser } from '../hooks/useUsers';

interface UserListProps {

searchTerm?: string;

}

export const UserList: React.FC<UserListProps> = ({ searchTerm }) => {

const [page, setPage] = useState(1);

const [showCreateForm, setShowCreateForm] = useState(false);

const {

data: usersResponse,

isLoading,

error,

refetch,

} = useUsers({

page,

limit: 20,

search: searchTerm,

});

const createUserMutation = useCreateUser();

const handleCreateUser = async (userData: CreateUserRequest) => {

try {

await createUserMutation.mutateAsync(userData);

setShowCreateForm(false);

} catch (error) {

console.error('Failed to create user:', error);

}

};

if (isLoading) {

return <div className="loading-spinner">Loading users...</div>;

}

if (error) {

return (

<div className="error-message">

<p>Failed to load users</p>

<button onClick={() => refetch()}>Retry</button>

</div>

);

}

const users = usersResponse?.data || [];

const pagination = usersResponse?.pagination;

return (

<div className="user-list">

<div className="user-list-header">

<h2>Users</h2>

<button onClick={() => setShowCreateForm(true)}>Add User</button>

</div>

{showCreateForm && (

<CreateUserForm

onSubmit={handleCreateUser}

onCancel={() => setShowCreateForm(false)}

isLoading={createUserMutation.isLoading}

/>

)}

<div className="user-grid">

{users.map((user) => (

<UserCard key={user.id} user={user} />

))}

</div>

{pagination && (

<Pagination

currentPage={pagination.page}

totalPages={pagination.pages}

onPageChange={setPage}

/>

)}

</div>

);

};

```

\*\*Backend API Implementation (Windsurf):\*\*

```javascript

// Express.js API with Validation and Error Handling

const express = require('express');

const { body, query, param, validationResult } = require('express-validator');

class UserController {

async getUsers(req, res, next) {

try {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

error: 'Validation failed',

details: errors.array(),

});

}

const { page = 1, limit = 20, search } = req.query;

const filters = {};

if (search) {

filters.$or = [

{ username: { $regex: search, $options: 'i' } },

{ email: { $regex: search, $options: 'i' } },

];

}

const users = await User.find(filters)

.limit(limit \* 1)

.skip((page - 1) \* limit)

.sort({ createdAt: -1 })

.select('-passwordHash');

const total = await User.countDocuments(filters);

res.json({

success: true,

data: users,

pagination: {

page: parseInt(page),

limit: parseInt(limit),

total,

pages: Math.ceil(total / limit),

},

});

} catch (error) {

next(error);

}

}

async getUserById(req, res, next) {

try {

const { id } = req.params;

const user = await User.findById(id).select('-passwordHash');

if (!user) {

return res.status(404).json({

success: false,

error: 'User not found',

});

}

res.json({

success: true,

data: user,

});

} catch (error) {

next(error);

}

}

async createUser(req, res, next) {

try {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

error: 'Validation failed',

details: errors.array(),

});

}

const { username, email, password } = req.body;

// Check if user already exists

const existingUser = await User.findOne({

$or: [{ email }, { username }],

});

if (existingUser) {

return res.status(409).json({

success: false,

error: 'User already exists',

});

}

// Hash password

const authService = new AuthService();

const passwordHash = await authService.hashPassword(password);

// Create user

const user = new User({

username,

email,

passwordHash,

roles: ['customer'],

isActive: true,

});

await user.save();

// Remove sensitive data

const userResponse = user.toObject();

delete userResponse.passwordHash;

res.status(201).json({

success: true,

data: userResponse,

message: 'User created successfully',

});

} catch (error) {

next(error);

}

}

}

// Route Definitions with Validation

const router = express.Router();

const userController = new UserController();

router.get(

'/users',

[

query('page').optional().isInt({ min: 1 }),

query('limit').optional().isInt({ min: 1, max: 100 }),

query('search').optional().isLength({ min: 1, max: 100 }),

],

authenticateJWT,

authorize(['admin', 'manager']),

userController.getUsers

);

router.get(

'/users/:id',

[param('id').isMongoId()],

authenticateJWT,

authorize(['admin', 'manager']),

userController.getUserById

);

router.post(

'/users',

[

body('username').isLength({ min: 3, max: 30 }).matches(/^[a-zA-Z0-9\_]+$/),

body('email').isEmail().normalizeEmail(),

body('password').isLength({ min: 8 }).matches(/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)/),

],

authenticateJWT,

authorize(['admin']),

userController.createUser

);

module.exports = router;

```

\*\*Real-time Integration with WebSockets:\*\*

```javascript

// WebSocket Integration (Backend)

const socketIo = require('socket.io');

class WebSocketService {

constructor(server) {

this.io = socketIo(server, {

cors: {

origin: process.env.FRONTEND\_URL,

methods: ['GET', 'POST'],

},

});

this.setupMiddleware();

this.setupEventHandlers();

}

setupMiddleware() {

this.io.use(async (socket, next) => {

try {

const token = socket.handshake.auth.token;

const authService = new AuthService();

const decoded = authService.verifyAccessToken(token);

socket.userId = decoded.userId;

socket.userRoles = decoded.roles;

next();

} catch (error) {

next(new Error('Authentication failed'));

}

});

}

setupEventHandlers() {

this.io.on('connection', (socket) => {

console.log(`User ${socket.userId} connected`);

// Join user-specific room

socket.join(`user:${socket.userId}`);

// Join role-based rooms

socket.userRoles.forEach(role => {

socket.join(`role:${role}`);

});

socket.on('disconnect', () => {

console.log(`User ${socket.userId} disconnected`);

});

});

}

emitToUser(userId, event, data) {

this.io.to(`user:${userId}`).emit(event, data);

}

emitToRole(role, event, data) {

this.io.to(`role:${role}`).emit(event, data);

}

broadcast(event, data) {

this.io.emit(event, data);

}

}

// Integration with API endpoints

class UserController {

constructor(websocketService) {

this.websocketService = websocketService;

}

async createUser(req, res, next) {

try {

// ... user creation logic ...

// Emit real-time update

this.websocketService.emitToRole('admin', 'user:created', {

user: userResponse,

timestamp: new Date(),

});

res.status(201).json({

success: true,

data: userResponse,

message: 'User created successfully',

});

} catch (error) {

next(error);

}

}

}

```

```typescript

// WebSocket Integration (Frontend)

import { io, Socket } from 'socket.io-client';

import { useEffect, useRef } from 'react';

class WebSocketClient {

private socket: Socket | null = null;

private reconnectAttempts = 0;

private maxReconnectAttempts = 5;

connect(token: string) {

this.socket = io(process.env.REACT\_APP\_WS\_URL, {

auth: { token },

autoConnect: true,

});

this.setupEventHandlers();

}

private setupEventHandlers() {

if (!this.socket) return;

this.socket.on('connect', () => {

console.log('WebSocket connected');

this.reconnectAttempts = 0;

});

this.socket.on('disconnect', (reason) => {

console.log('WebSocket disconnected:', reason);

if (reason === 'io server disconnect') {

// Server initiated disconnect, reconnect manually

this.reconnect();

}

});

this.socket.on('connect\_error', (error) => {

console.error('WebSocket connection error:', error);

this.handleReconnect();

});

}

private handleReconnect() {

if (this.reconnectAttempts < this.maxReconnectAttempts) {

this.reconnectAttempts++;

setTimeout(() => this.reconnect(), 1000 \* this.reconnectAttempts);

}

}

private reconnect() {

this.socket?.connect();

}

on(event: string, callback: (...args: any[]) => void) {

this.socket?.on(event, callback);

}

off(event: string, callback?: (...args: any[]) => void) {

this.socket?.off(event, callback);

}

emit(event: string, data: any) {

this.socket?.emit(event, data);

}

disconnect() {

this.socket?.disconnect();

this.socket = null;

}

}

// React Hook for WebSocket

export function useWebSocket() {

const wsClient = useRef<WebSocketClient | null>(null);

useEffect(() => {

const token = localStorage.getItem('authToken');

if (token) {

wsClient.current = new WebSocketClient();

wsClient.current.connect(token);

}

return () => {

wsClient.current?.disconnect();

};

}, []);

const subscribe = (event: string, callback: (...args: any[]) => void) => {

wsClient.current?.on(event, callback);

return () => {

wsClient.current?.off(event, callback);

};

};

const emit = (event: string, data: any) => {

wsClient.current?.emit(event, data);

};

return { subscribe, emit };

}

```

\*\*Error Handling & Retry Logic:\*\*

```typescript

// Advanced Error Handling

interface ApiError {

message: string;

code: string;

status: number;

details?: any;

}

class ApiErrorHandler {

static handle(error: any): ApiError {

if (error.response) {

// Server responded with error status

return {

message: error.response.data?.error || 'Server error',

code: error.response.data?.code || 'SERVER\_ERROR',

status: error.response.status,

details: error.response.data?.details,

};

} else if (error.request) {

// Network error

return {

message: 'Network error. Please check your connection.',

code: 'NETWORK\_ERROR',

status: 0,

};

} else {

// Other error

return {

message: error.message || 'An unexpected error occurred',

code: 'UNKNOWN\_ERROR',

status: 0,

};

}

}

static isRetryable(error: ApiError): boolean {

return (

error.status >= 500 || // Server errors

error.code === 'NETWORK\_ERROR' ||

error.status === 429 // Rate limited

);

}

}

// Retry Logic with Exponential Backoff

async function withRetry<T>(

operation: () => Promise<T>,

maxAttempts: number = 3,

baseDelay: number = 1000

): Promise<T> {

let lastError: any;

for (let attempt = 1; attempt <= maxAttempts; attempt++) {

try {

return await operation();

} catch (error) {

lastError = error;

const apiError = ApiErrorHandler.handle(error);

if (attempt === maxAttempts || !ApiErrorHandler.isRetryable(apiError)) {

throw apiError;

}

// Exponential backoff with jitter

const delay = baseDelay \* Math.pow(2, attempt - 1);

const jitter = Math.random() \* 0.1 \* delay;

await new Promise(resolve => setTimeout(resolve, delay + jitter));

}

}

throw lastError;

}

```

\*\*AWS Integration:\*\*

- API Gateway: [REQUEST\_VALIDATION/CORS\_CONFIGURATION/THROTTLING/MONITORING]

- Lambda Functions: [API\_HANDLERS/COLD\_START\_OPTIMIZATION/ERROR\_HANDLING]

- CloudFront: [API\_CACHING/EDGE\_OPTIMIZATION/SECURITY\_HEADERS]

- WebSocket API: [REAL\_TIME\_COMMUNICATION/CONNECTION\_MANAGEMENT/SCALING]

- Monitoring: [CLOUDWATCH\_METRICS/X\_RAY\_TRACING/CUSTOM\_DASHBOARDS]

Provide complete full-stack integration with API client implementation, React Query integration, WebSocket support, error handling, and AWS deployment configuration.

```

---

## DevOps & Deployment Templates

### 17. CI/CD Pipeline Implementation

```

Act as a DevOps engineer specializing in CI/CD for lovable.dev and windsurf. Create a comprehensive CI/CD pipeline for [PROJECT\_NAME] that:

\*\*Pipeline Architecture:\*\*

- Version Control: [GIT\_WORKFLOW/BRANCHING\_STRATEGY/MERGE\_POLICIES]

- CI/CD Platform: [GITHUB\_ACTIONS/AWS\_CODEPIPELINE/JENKINS/CUSTOM\_SOLUTION]

- Environment Strategy: [DEV/STAGING/PROD/FEATURE\_BRANCHES/PREVIEW\_ENVIRONMENTS]

- Deployment Strategy: [BLUE\_GREEN/CANARY/ROLLING/IMMUTABLE\_DEPLOYMENTS]

- Rollback Strategy: [AUTOMATED\_ROLLBACK/MANUAL\_ROLLBACK/HEALTH\_CHECK\_TRIGGERS]

\*\*GitHub Actions CI/CD Pipeline:\*\*

```yaml

# .github/workflows/ci-cd.yml

name: CI/CD Pipeline

on:

push:

branches: [main, develop]

pull\_request:

branches: [main, develop]

env:

NODE\_VERSION: '18'

AWS\_REGION: ${{ secrets.AWS\_REGION }}

jobs:

# Code Quality & Testing

quality-checks:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: ${{ env.NODE\_VERSION }}

cache: 'npm'

- name: Install dependencies

run: npm ci

- name: Run linting

run: npm run lint

- name: Run type checking

run: npm run type-check

- name: Run unit tests

run: npm run test:unit

- name: Run integration tests

run: npm run test:integration

- name: Generate test coverage

run: npm run test:coverage

- name: Upload coverage to Codecov

uses: codecov/codecov-action@v3

with:

file: ./coverage/lcov.info

# Security Scanning

security-scan:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Run dependency vulnerability scan

run: npm audit --audit-level=moderate

- name: Run SAST scan

uses: github/super-linter@v4

env:

DEFAULT\_BRANCH: main

GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN }}

- name: Run container security scan

uses: aquasecurity/trivy-action@master

with:

scan-type: 'fs'

scan-ref: '.'

# Frontend Build & Deploy (Lovable.dev)

frontend-deploy:

runs-on: ubuntu-latest

needs: [quality-checks, security-scan]

if: github.ref == 'refs/heads/main'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: ${{ env.NODE\_VERSION }}

cache: 'npm'

- name: Install dependencies

working-directory: ./frontend

run: npm ci

- name: Build frontend

working-directory: ./frontend

run: npm run build

env:

REACT\_APP\_API\_URL: ${{ secrets.API\_URL\_PROD }}

REACT\_APP\_ENV: production

- name: Run E2E tests

working-directory: ./frontend

run: npm run test:e2e

- name: Configure AWS credentials

uses: aws-actions/configure-aws-credentials@v4

with:

aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

aws-region: ${{ env.AWS\_REGION }}

- name: Deploy to AWS Amplify

run: |

aws amplify start-job \

--app-id ${{ secrets.AMPLIFY\_APP\_ID }} \

--branch-name main \

--job-type RELEASE

- name: Wait for deployment

run: |

aws amplify get-job \

--app-id ${{ secrets.AMPLIFY\_APP\_ID }} \

--branch-name main \

--job-id $(aws amplify list-jobs --app-id ${{ secrets.AMPLIFY\_APP\_ID }} --branch-name main --query 'jobSummaries[0].jobId' --output text) \

--query 'job.status' \

--output text

- name: Run smoke tests

run: |

curl -f ${{ secrets.FRONTEND\_URL }}/health || exit 1

# Backend Build & Deploy (Windsurf)

backend-deploy:

runs-on: ubuntu-latest

needs: [quality-checks, security-scan]

if: github.ref == 'refs/heads/main'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: ${{ env.NODE\_VERSION }}

cache: 'npm'

- name: Install dependencies

working-directory: ./backend

run: npm ci

- name: Build backend

working-directory: ./backend

run: npm run build

- name: Run database migrations

working-directory: ./backend

run: npm run migrate

env:

DATABASE\_URL: ${{ secrets.DATABASE\_URL\_STAGING }}

- name: Configure AWS credentials

uses: aws-actions/configure-aws-credentials@v4

with:

aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

aws-region: ${{ env.AWS\_REGION }}

- name: Deploy to AWS Lambda

run: |

# Package Lambda function

zip -r function.zip . -x "\*.git\*" "node\_modules/.cache/\*" "tests/\*"

# Update Lambda function

aws lambda update-function-code \

--function-name ${{ secrets.LAMBDA\_FUNCTION\_NAME }} \

--zip-file fileb://function.zip

- name: Update Lambda environment variables

run: |

aws lambda update-function-configuration \

--function-name ${{ secrets.LAMBDA\_FUNCTION\_NAME }} \

--environment Variables="{

NODE\_ENV=production,

DATABASE\_URL=${{ secrets.DATABASE\_URL\_PROD }},

JWT\_SECRET=${{ secrets.JWT\_SECRET }},

REDIS\_URL=${{ secrets.REDIS\_URL }}

}"

- name: Run API health check

run: |

sleep 30 # Wait for deployment

curl -f ${{ secrets.API\_URL }}/health || exit 1

# Database Migration

database-migrate:

runs-on: ubuntu-latest

needs: [quality-checks]

if: github.ref == 'refs/heads/main'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: ${{ env.NODE\_VERSION }}

- name: Install dependencies

run: npm ci

- name: Run database migrations

run: npm run migrate:prod

env:

DATABASE\_URL: ${{ secrets.DATABASE\_URL\_PROD }}

- name: Verify migration

run: npm run migrate:status

env:

DATABASE\_URL: ${{ secrets.DATABASE\_URL\_PROD }}

# Performance Testing

performance-test:

runs-on: ubuntu-latest

needs: [frontend-deploy, backend-deploy]

if: github.ref == 'refs/heads/main'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Run Lighthouse CI

uses: treosh/lighthouse-ci-action@v10

with:

urls: |

${{ secrets.FRONTEND\_URL }}

${{ secrets.FRONTEND\_URL }}/dashboard

uploadArtifacts: true

temporaryPublicStorage: true

- name: Run load tests

run: |

npx artillery run tests/load-test.yml \

--target ${{ secrets.API\_URL }} \

--output report.json

- name: Generate load test report

run: npx artillery report report.json

# Staging Deployment

staging-deploy:

runs-on: ubuntu-latest

needs: [quality-checks, security-scan]

if: github.ref == 'refs/heads/develop'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Deploy to staging

run: |

echo "Deploying to staging environment..."

# Similar deployment steps but for staging environment

# Feature Branch Preview

preview-deploy:

runs-on: ubuntu-latest

if: github.event\_name == 'pull\_request'

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Deploy preview environment

run: |

echo "Creating preview environment for PR #${{ github.event.number }}"

# Deploy to preview environment with unique URL

- name: Comment PR with preview URL

uses: actions/github-script@v6

with:

script: |

github.rest.issues.createComment({

issue\_number: context.issue.number,

owner: context.repo.owner,

repo: context.repo.repo,

body: '🚀 Preview environment deployed: https://pr-${{ github.event.number }}.preview.example.com'

})

```

\*\*AWS CodePipeline Alternative:\*\*

```yaml

# buildspec.yml for AWS CodeBuild

version: 0.2

phases:

install:

runtime-versions:

nodejs: 18

commands:

- echo Installing dependencies...

- npm ci

pre\_build:

commands:

- echo Running pre-build tasks...

- npm run lint

- npm run test:unit

- npm run test:integration

build:

commands:

- echo Building the application...

- npm run build

- echo Build completed on `date`

post\_build:

commands:

- echo Running post-build tasks...

- npm run test:e2e

- echo Deploying to S3...

- aws s3 sync dist/ s3://$S3\_BUCKET --delete

- echo Invalidating CloudFront...

- aws cloudfront create-invalidation --distribution-id $CLOUDFRONT\_ID --paths "/\*"

artifacts:

files:

- '\*\*/\*'

base-directory: dist

name: BuildArtifact

cache:

paths:

- node\_modules/\*\*/\*

```

\*\*Infrastructure as Code (Terraform):\*\*

```hcl

# infrastructure/main.tf

terraform {

required\_version = ">= 1.0"

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

backend "s3" {

bucket = "my-terraform-state-bucket"

key = "ci-cd/terraform.tfstate"

region = "us-west-2"

}

}

provider "aws" {

region = var.aws\_region

}

# S3 Bucket for Frontend

resource "aws\_s3\_bucket" "frontend" {

bucket = "${var.project\_name}-frontend-${var.environment}"

}

resource "aws\_s3\_bucket\_public\_access\_block" "frontend" {

bucket = aws\_s3\_bucket.frontend.id

block\_public\_acls = false

block\_public\_policy = false

ignore\_public\_acls = false

restrict\_public\_buckets = false

}

resource "aws\_s3\_bucket\_website\_configuration" "frontend" {

bucket = aws\_s3\_bucket.frontend.id

index\_document {

suffix = "index.html"

}

error\_document {

key = "error.html"

}

}

# CloudFront Distribution

resource "aws\_cloudfront\_distribution" "frontend" {

origin {

domain\_name = aws\_s3\_bucket.frontend.bucket\_regional\_domain\_name

origin\_id = "S3-${aws\_s3\_bucket.frontend.id}"

s3\_origin\_config {

origin\_access\_identity = aws\_cloudfront\_origin\_access\_identity.frontend.cloudfront\_access\_identity\_path

}

}

enabled = true

is\_ipv6\_enabled = true

default\_root\_object = "index.html"

default\_cache\_behavior {

allowed\_methods = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]

cached\_methods = ["GET", "HEAD"]

target\_origin\_id = "S3-${aws\_s3\_bucket.frontend.id}"

compress = true

viewer\_protocol\_policy = "redirect-to-https"

forwarded\_values {

query\_string = false

cookies {

forward = "none"

}

}

min\_ttl = 0

default\_ttl = 3600

max\_ttl = 86400

}

price\_class = "PriceClass\_100"

restrictions {

geo\_restriction {

restriction\_type = "none"

}

}

viewer\_certificate {

cloudfront\_default\_certificate = true

}

tags = {

Name = "${var.project\_name}-frontend"

Environment = var.environment

}

}

# Lambda Function for Backend

resource "aws\_lambda\_function" "backend" {

filename = "backend.zip"

function\_name = "${var.project\_name}-backend-${var.environment}"

role = aws\_iam\_role.lambda\_role.arn

handler = "index.handler"

source\_code\_hash = filebase64sha256("backend.zip")

runtime = "nodejs18.x"

timeout = 30

memory\_size = 512

environment {

variables = {

NODE\_ENV = var.environment

DATABASE\_URL = var.database\_url

JWT\_SECRET = var.jwt\_secret

}

}

tags = {

Name = "${var.project\_name}-backend"

Environment = var.environment

}

}

# API Gateway

resource "aws\_api\_gateway\_rest\_api" "backend" {

name = "${var.project\_name}-api-${var.environment}"

description = "API Gateway for ${var.project\_name}"

endpoint\_configuration {

types = ["REGIONAL"]

}

}

resource "aws\_api\_gateway\_resource" "proxy" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.backend.id

parent\_id = aws\_api\_gateway\_rest\_api.backend.root\_resource\_id

path\_part = "{proxy+}"

}

resource "aws\_api\_gateway\_method" "proxy" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.backend.id

resource\_id = aws\_api\_gateway\_resource.proxy.id

http\_method = "ANY"

authorization = "NONE"

}

resource "aws\_api\_gateway\_integration" "lambda" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.backend.id

resource\_id = aws\_api\_gateway\_method.proxy.resource\_id

http\_method = aws\_api\_gateway\_method.proxy.http\_method

integration\_http\_method = "POST"

type = "AWS\_PROXY"

uri = aws\_lambda\_function.backend.invoke\_arn

}

# RDS Database

resource "aws\_db\_instance" "main" {

identifier = "${var.project\_name}-db-${var.environment}"

engine = "postgres"

engine\_version = "15.4"

instance\_class = var.db\_instance\_class

allocated\_storage = 20

max\_allocated\_storage = 100

storage\_type = "gp2"

storage\_encrypted = true

db\_name = var.database\_name

username = var.database\_username

password = var.database\_password

vpc\_security\_group\_ids = [aws\_security\_group.rds.id]

db\_subnet\_group\_name = aws\_db\_subnet\_group.main.name

backup\_retention\_period = 7

backup\_window = "03:00-04:00"

maintenance\_window = "sun:04:00-sun:05:00"

skip\_final\_snapshot = var.environment != "production"

deletion\_protection = var.environment == "production"

tags = {

Name = "${var.project\_name}-db"

Environment = var.environment

}

}

```

\*\*Monitoring & Alerting:\*\*

```yaml

# monitoring/cloudwatch-alarms.yml

Resources:

HighErrorRate:

Type: AWS::CloudWatch::Alarm

Properties:

AlarmName: !Sub "${ProjectName}-high-error-rate"

AlarmDescription: "High error rate detected"

MetricName: Errors

Namespace: AWS/Lambda

Statistic: Sum

Period: 300

EvaluationPeriods: 2

Threshold: 10

ComparisonOperator: GreaterThanThreshold

AlarmActions:

- !Ref SNSTopicArn

HighLatency:

Type: AWS::CloudWatch::Alarm

Properties:

AlarmName: !Sub "${ProjectName}-high-latency"

AlarmDescription: "High response latency detected"

MetricName: Duration

Namespace: AWS/Lambda

Statistic: Average

Period: 300

EvaluationPeriods: 2

Threshold: 5000

ComparisonOperator: GreaterThanThreshold

AlarmActions:

- !Ref SNSTopicArn

LowDiskSpace:

Type: AWS::CloudWatch::Alarm

Properties:

AlarmName: !Sub "${ProjectName}-low-disk-space"

AlarmDescription: "RDS storage space is running low"

MetricName: FreeStorageSpace

Namespace: AWS/RDS

Statistic: Average

Period: 300

EvaluationPeriods: 1

Threshold: 2000000000 # 2GB in bytes

ComparisonOperator: LessThanThreshold

AlarmActions:

- !Ref SNSTopicArn

```

\*\*Deployment Scripts:\*\*

```bash

#!/bin/bash

# scripts/deploy.sh

set -e

PROJECT\_NAME="my-project"

ENVIRONMENT=${1:-staging}

AWS\_REGION=${AWS\_REGION:-us-west-2}

echo "🚀 Starting deployment for $PROJECT\_NAME to $ENVIRONMENT"

# Build frontend

echo "📦 Building frontend..."

cd frontend

npm ci

npm run build

cd ..

# Build backend

echo "📦 Building backend..."

cd backend

npm ci

npm run build

cd ..

# Deploy infrastructure

echo "🏗️ Deploying infrastructure..."

cd infrastructure

terraform init

terraform plan -var="environment=$ENVIRONMENT"

terraform apply -var="environment=$ENVIRONMENT" -auto-approve

cd ..

# Deploy frontend to S3

echo "🌐 Deploying frontend..."

aws s3 sync frontend/build/ s3://$PROJECT\_NAME-frontend-$ENVIRONMENT --delete

aws cloudfront create-invalidation \

--distribution-id $(terraform output -raw cloudfront\_distribution\_id) \

--paths "/\*"

# Deploy backend to Lambda

echo "⚡ Deploying backend..."

cd backend

zip -r ../backend.zip . -x "node\_modules/.cache/\*" "tests/\*"

cd ..

aws lambda update-function-code \

--function-name $PROJECT\_NAME-backend-$ENVIRONMENT \

--zip-file fileb://backend.zip

# Run database migrations

echo "🗄️ Running database migrations..."

DATABASE\_URL=$(terraform output -raw database\_url) npm run migrate

# Run health checks

echo "🏥 Running health checks..."

sleep 30

FRONTEND\_URL=$(terraform output -raw frontend\_url)

API\_URL=$(terraform output -raw api\_url)

curl -f "$FRONTEND\_URL/health" || { echo "Frontend health check failed"; exit 1; }

curl -f "$API\_URL/health" || { echo "API health check failed"; exit 1; }

echo "✅ Deployment completed successfully!"

echo "Frontend: $FRONTEND\_URL"

echo "API: $API\_URL"

```

\*\*Cost Optimization:\*\*

- Resource Right-sizing: [LAMBDA\_MEMORY\_OPTIMIZATION/RDS\_INSTANCE\_SIZING/S3\_STORAGE\_CLASS]

- Usage Monitoring: [COST\_ALERTS/BUDGET\_CONTROLS/USAGE\_REPORTS]

- Efficient Deployments: [INCREMENTAL\_BUILDS/ASSET\_CACHING/COMPRESSION]

- Environment Management: [EPHEMERAL\_ENVIRONMENTS/RESOURCE\_CLEANUP/SCHEDULED\_SHUTDOWNS]

Provide complete CI/CD pipeline with GitHub Actions workflows, infrastructure as code, monitoring setup, deployment scripts, and cost optimization strategies.

```

### 18. Infrastructure as Code Implementation

```

Act as an infrastructure architect specializing in AWS and Terraform for windsurf. Create a comprehensive Infrastructure as Code solution for [PROJECT\_NAME] that:

\*\*Infrastructure Architecture:\*\*

- Cloud Provider: [AWS/AZURE/GCP/MULTI\_CLOUD]

- IaC Tool: [TERRAFORM/CLOUDFORMATION/CDK/PULUMI]

- State Management: [REMOTE\_STATE/STATE\_LOCKING/VERSIONING/BACKUP]

- Environment Strategy: [DEV/STAGING/PROD/FEATURE\_ENVIRONMENTS]

- Module Organization: [REUSABLE\_MODULES/ENVIRONMENT\_CONFIGS/SHARED\_RESOURCES]

\*\*Terraform Configuration:\*\*

```hcl

# main.tf - Root Configuration

terraform {

required\_version = ">= 1.0"

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

random = {

source = "hashicorp/random"

version = "~> 3.1"

}

}

backend "s3" {

bucket = "terraform-state-bucket"

key = "infrastructure/terraform.tfstate"

region = "us-west-2"

encrypt = true

dynamodb\_table = "terraform-state-lock"

}

}

# Provider Configuration

provider "aws" {

region = var.aws\_region

default\_tags {

tags = {

Project = var.project\_name

Environment = var.environment

ManagedBy = "terraform"

Owner = var.owner

}

}

}

# Data Sources

data "aws\_availability\_zones" "available" {

state = "available"

}

data "aws\_caller\_identity" "current" {}

# Local Values

locals {

common\_tags = {

Project = var.project\_name

Environment = var.environment

ManagedBy = "terraform"

}

vpc\_cidr = "10.0.0.0/16"

azs = slice(data.aws\_availability\_zones.available.names, 0, 2)

}

# VPC Module

module "vpc" {

source = "./modules/vpc"

project\_name = var.project\_name

environment = var.environment

vpc\_cidr = local.vpc\_cidr

azs = local.azs

enable\_nat\_gateway = var.environment == "production"

enable\_vpn\_gateway = false

tags = local.common\_tags

}

# Security Module

module "security" {

source = "./modules/security"

project\_name = var.project\_name

environment = var.environment

vpc\_id = module.vpc.vpc\_id

tags = local.common\_tags

}

# Database Module

module "database" {

source = "./modules/database"

project\_name = var.project\_name

environment = var.environment

vpc\_id = module.vpc.vpc\_id

private\_subnet\_ids = module.vpc.private\_subnet\_ids

security\_group\_ids = [module.security.rds\_security\_group\_id]

instance\_class = var.db\_instance\_class

allocated\_storage = var.db\_allocated\_storage

database\_name = var.database\_name

database\_username = var.database\_username

database\_password = var.database\_password

backup\_retention\_period = var.environment == "production" ? 7 : 1

multi\_az = var.environment == "production"

deletion\_protection = var.environment == "production"

tags = local.common\_tags

}

# Application Module

module "application" {

source = "./modules/application"

project\_name = var.project\_name

environment = var.environment

vpc\_id = module.vpc.vpc\_id

private\_subnet\_ids = module.vpc.private\_subnet\_ids

public\_subnet\_ids = module.vpc.public\_subnet\_ids

security\_group\_ids = [module.security.lambda\_security\_group\_id]

database\_url = module.database.connection\_string

tags = local.common\_tags

}

# Frontend Module

module "frontend" {

source = "./modules/frontend"

project\_name = var.project\_name

environment = var.environment

domain\_name = var.domain\_name

api\_url = module.application.api\_gateway\_url

tags = local.common\_tags

}

# Monitoring Module

module "monitoring" {

source = "./modules/monitoring"

project\_name = var.project\_name

environment = var.environment

lambda\_function\_name = module.application.lambda\_function\_name

rds\_instance\_id = module.database.instance\_id

cloudfront\_distribution\_id = module.frontend.cloudfront\_distribution\_id

notification\_email = var.notification\_email

tags = local.common\_tags

}

```

\*\*VPC Module:\*\*

```hcl

# modules/vpc/main.tf

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

enable\_dns\_hostnames = true

enable\_dns\_support = true

tags = merge(var.tags, {

Name = "${var.project\_name}-vpc-${var.environment}"

})

}

# Internet Gateway

resource "aws\_internet\_gateway" "main" {

vpc\_id = aws\_vpc.main.id

tags = merge(var.tags, {

Name = "${var.project\_name}-igw-${var.environment}"

})

}

# Public Subnets

resource "aws\_subnet" "public" {

count = length(var.azs)

vpc\_id = aws\_vpc.main.id

cidr\_block = cidrsubnet(var.vpc\_cidr, 8, count.index)

availability\_zone = var.azs[count.index]

map\_public\_ip\_on\_launch = true

tags = merge(var.tags, {

Name = "${var.project\_name}-public-${var.azs[count.index]}-${var.environment}"

Type = "public"

})

}

# Private Subnets

resource "aws\_subnet" "private" {

count = length(var.azs)

vpc\_id = aws\_vpc.main.id

cidr\_block = cidrsubnet(var.vpc\_cidr, 8, count.index + 10)

availability\_zone = var.azs[count.index]

tags = merge(var.tags, {

Name = "${var.project\_name}-private-${var.azs[count.index]}-${var.environment}"

Type = "private"

})

}

# Database Subnets

resource "aws\_subnet" "database" {

count = length(var.azs)

vpc\_id = aws\_vpc.main.id

cidr\_block = cidrsubnet(var.vpc\_cidr, 8, count.index + 20)

availability\_zone = var.azs[count.index]

tags = merge(var.tags, {

Name = "${var.project\_name}-database-${var.azs[count.index]}-${var.environment}"

Type = "database"

})

}

# NAT Gateways

resource "aws\_eip" "nat" {

count = var.enable\_nat\_gateway ? length(var.azs) : 0

domain = "vpc"

depends\_on = [aws\_internet\_gateway.main]

tags = merge(var.tags, {

Name = "${var.project\_name}-nat-eip-${count.index + 1}-${var.environment}"

})

}

resource "aws\_nat\_gateway" "main" {

count = var.enable\_nat\_gateway ? length(var.azs) : 0

allocation\_id = aws\_eip.nat[count.index].id

subnet\_id = aws\_subnet.public[count.index].id

tags = merge(var.tags, {

Name = "${var.project\_name}-nat-${count.index + 1}-${var.environment}"

})

depends\_on = [aws\_internet\_gateway.main]

}

# Route Tables

resource "aws\_route\_table" "public" {

vpc\_id = aws\_vpc.main.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.main.id

}

tags = merge(var.tags, {

Name = "${var.project\_name}-public-rt-${var.environment}"

})

}

resource "aws\_route\_table" "private" {

count = length(var.azs)

vpc\_id = aws\_vpc.main.id

dynamic "route" {

for\_each = var.enable\_nat\_gateway ? [1] : []

content {

cidr\_block = "0.0.0.0/0"

nat\_gateway\_id = aws\_nat\_gateway.main[count.index].id

}

}

tags = merge(var.tags, {

Name = "${var.project\_name}-private-rt-${count.index + 1}-${var.environment}"

})

}

# Route Table Associations

resource "aws\_route\_table\_association" "public" {

count = length(aws\_subnet.public)

subnet\_id = aws\_subnet.public[count.index].id

route\_table\_id = aws\_route\_table.public.id

}

resource "aws\_route\_table\_association" "private" {

count = length(aws\_subnet.private)

subnet\_id = aws\_subnet.private[count.index].id

route\_table\_id = aws\_route\_table.private[count.index].id

}

# Database Subnet Group

resource "aws\_db\_subnet\_group" "main" {

name = "${var.project\_name}-db-subnet-group-${var.environment}"

subnet\_ids = aws\_subnet.database[\*].id

tags = merge(var.tags, {

Name = "${var.project\_name}-db-subnet-group-${var.environment}"

})

}

```

\*\*Database Module:\*\*

```hcl

# modules/database/main.tf

resource "random\_password" "master" {

count = var.database\_password == null ? 1 : 0

length = 16

special = true

}

resource "aws\_db\_parameter\_group" "main" {

family = "postgres15"

name = "${var.project\_name}-db-params-${var.environment}"

parameter {

name = "log\_statement"

value = "all"

}

parameter {

name = "log\_min\_duration\_statement"

value = "1000"

}

parameter {

name = "shared\_preload\_libraries"

value = "pg\_stat\_statements"

}

tags = var.tags

}

resource "aws\_db\_instance" "main" {

identifier = "${var.project\_name}-db-${var.environment}"

# Engine

engine = "postgres"

engine\_version = "15.4"

instance\_class = var.instance\_class

# Storage

allocated\_storage = var.allocated\_storage

max\_allocated\_storage = var.allocated\_storage \* 2

storage\_type = "gp3"

storage\_encrypted = true

# Database

db\_name = var.database\_name

username = var.database\_username

password = coalesce(var.database\_password, try(random\_password.master[0].result, null))

# Network

vpc\_security\_group\_ids = var.security\_group\_ids

db\_subnet\_group\_name = var.db\_subnet\_group\_name

# Backup

backup\_retention\_period = var.backup\_retention\_period

backup\_window = "03:00-04:00"

copy\_tags\_to\_snapshot = true

# Maintenance

maintenance\_window = "sun:04:00-sun:05:00"

auto\_minor\_version\_upgrade = var.environment == "production" ? false : true

# High Availability

multi\_az = var.multi\_az

# Parameter Group

parameter\_group\_name = aws\_db\_parameter\_group.main.name

# Monitoring

monitoring\_interval = 60

monitoring\_role\_arn = aws\_iam\_role.rds\_monitoring.arn

performance\_insights\_enabled = true

performance\_insights\_retention\_period = var.environment == "production" ? 7 : 7

# Deletion Protection

deletion\_protection = var.deletion\_protection

skip\_final\_snapshot = !var.deletion\_protection

tags = merge(var.tags, {

Name = "${var.project\_name}-database-${var.environment}"

})

}

# Enhanced Monitoring IAM Role

resource "aws\_iam\_role" "rds\_monitoring" {

name = "${var.project\_name}-rds-monitoring-${var.environment}"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "monitoring.rds.amazonaws.com"

}

}

]

})

tags = var.tags

}

resource "aws\_iam\_role\_policy\_attachment" "rds\_monitoring" {

role = aws\_iam\_role.rds\_monitoring.name

policy\_arn = "arn:aws:iam::aws:policy/service-role/AmazonRDSEnhancedMonitoringRole"

}

# Secrets Manager for Database Credentials

resource "aws\_secretsmanager\_secret" "db\_credentials" {

name = "${var.project\_name}-db-credentials-${var.environment}"

description = "Database credentials for ${var.project\_name}"

tags = var.tags

}

resource "aws\_secretsmanager\_secret\_version" "db\_credentials" {

secret\_id = aws\_secretsmanager\_secret.db\_credentials.id

secret\_string = jsonencode({

username = aws\_db\_instance.main.username

password = aws\_db\_instance.main.password

engine = "postgres"

host = aws\_db\_instance.main.endpoint

port = aws\_db\_instance.main.port

dbname = aws\_db\_instance.main.db\_name

})

}

```

\*\*Application Module:\*\*

```hcl

# modules/application/main.tf

# Lambda Function

resource "aws\_lambda\_function" "main" {

filename = data.archive\_file.lambda\_zip.output\_path

function\_name = "${var.project\_name}-api-${var.environment}"

role = aws\_iam\_role.lambda\_role.arn

handler = "index.handler"

source\_code\_hash = data.archive\_file.lambda\_zip.output\_base64sha256

runtime = "nodejs18.x"

timeout = 30

memory\_size = var.lambda\_memory\_size

vpc\_config {

subnet\_ids = var.private\_subnet\_ids

security\_group\_ids = var.security\_group\_ids

}

environment {

variables = {

NODE\_ENV = var.environment

DATABASE\_URL = var.database\_url

JWT\_SECRET = var.jwt\_secret

REDIS\_URL = aws\_elasticache\_cluster.redis.cache\_nodes[0].address

}

}

dead\_letter\_config {

target\_arn = aws\_sqs\_queue.dlq.arn

}

tags = var.tags

}

# Lambda ZIP Archive

data "archive\_file" "lambda\_zip" {

type = "zip"

output\_path = "/tmp/lambda\_function.zip"

source\_dir = "${path.module}/../../backend/dist"

}

# Lambda IAM Role

resource "aws\_iam\_role" "lambda\_role" {

name = "${var.project\_name}-lambda-role-${var.environment}"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "lambda.amazonaws.com"

}

}

]

})

tags = var.tags

}

# Lambda IAM Policy

resource "aws\_iam\_role\_policy\_attachment" "lambda\_basic" {

policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

role = aws\_iam\_role.lambda\_role.name

}

resource "aws\_iam\_role\_policy\_attachment" "lambda\_vpc" {

policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

role = aws\_iam\_role.lambda\_role.name

}

resource "aws\_iam\_role\_policy" "lambda\_custom" {

name = "${var.project\_name}-lambda-custom-${var.environment}"

role = aws\_iam\_role.lambda\_role.id

policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Effect = "Allow"

Action = [

"secretsmanager:GetSecretValue",

"sqs:SendMessage",

"elasticache:\*"

]

Resource = [

"arn:aws:secretsmanager:\*:\*:secret:${var.project\_name}-\*",

aws\_sqs\_queue.dlq.arn,

aws\_elasticache\_cluster.redis.arn

]

}

]

})

}

# API Gateway

resource "aws\_api\_gateway\_rest\_api" "main" {

name = "${var.project\_name}-api-${var.environment}"

description = "API Gateway for ${var.project\_name}"

endpoint\_configuration {

types = ["REGIONAL"]

}

tags = var.tags

}

# API Gateway Deployment

resource "aws\_api\_gateway\_deployment" "main" {

depends\_on = [

aws\_api\_gateway\_method.proxy,

aws\_api\_gateway\_integration.lambda

]

rest\_api\_id = aws\_api\_gateway\_rest\_api.main.id

stage\_name = var.environment

lifecycle {

create\_before\_destroy = true

}

}

# API Gateway Resource

resource "aws\_api\_gateway\_resource" "proxy" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.main.id

parent\_id = aws\_api\_gateway\_rest\_api.main.root\_resource\_id

path\_part = "{proxy+}"

}

# API Gateway Method

resource "aws\_api\_gateway\_method" "proxy" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.main.id

resource\_id = aws\_api\_gateway\_resource.proxy.id

http\_method = "ANY"

authorization = "NONE"

}

# API Gateway Integration

resource "aws\_api\_gateway\_integration" "lambda" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.main.id

resource\_id = aws\_api\_gateway\_method.proxy.resource\_id

http\_method = aws\_api\_gateway\_method.proxy.http\_method

integration\_http\_method = "POST"

type = "AWS\_PROXY"

uri = aws\_lambda\_function.main.invoke\_arn

}

# Lambda Permission for API Gateway

resource "aws\_lambda\_permission" "api\_gateway" {

statement\_id = "AllowExecutionFromAPIGateway"

action = "lambda:InvokeFunction"

function\_name = aws\_lambda\_function.main.function\_name

principal = "apigateway.amazonaws.com"

source\_arn = "${aws\_api\_gateway\_rest\_api.main.execution\_arn}/\*/\*"

}

# Redis Cache

resource "aws\_elasticache\_subnet\_group" "redis" {

name = "${var.project\_name}-redis-subnet-${var.environment}"

subnet\_ids = var.private\_subnet\_ids

tags = var.tags

}

resource "aws\_elasticache\_cluster" "redis" {

cluster\_id = "${var.project\_name}-redis-${var.environment}"

engine = "redis"

node\_type = var.redis\_node\_type

num\_cache\_nodes = 1

parameter\_group\_name = "default.redis7"

port = 6379

subnet\_group\_name = aws\_elasticache\_subnet\_group.redis.name

security\_group\_ids = var.security\_group\_ids

tags = var.tags

}

# Dead Letter Queue

resource "aws\_sqs\_queue" "dlq" {

name = "${var.project\_name}-dlq-${var.environment}"

message\_retention\_seconds = 1209600 # 14 days

tags = var.tags

}

```

\*\*Environment-Specific Configurations:\*\*

```hcl

# environments/production/terraform.tfvars

project\_name = "my-project"

environment = "production"

aws\_region = "us-west-2"

owner = "devops-team"

# Database Configuration

db\_instance\_class = "db.t3.medium"

db\_allocated\_storage = 100

database\_name = "myproject\_prod"

database\_username = "admin"

# Lambda Configuration

lambda\_memory\_size = 1024

# Redis Configuration

redis\_node\_type = "cache.t3.micro"

# Domain Configuration

domain\_name = "myproject.com"

# Monitoring

notification\_email = "alerts@myproject.com"

# Feature Flags

enable\_monitoring = true

enable\_backup = true

```

```hcl

# environments/staging/terraform.tfvars

project\_name = "my-project"

environment = "staging"

aws\_region = "us-west-2"

owner = "devops-team"

# Database Configuration

db\_instance\_class = "db.t3.micro"

db\_allocated\_storage = 20

database\_name = "myproject\_staging"

database\_username = "admin"

# Lambda Configuration

lambda\_memory\_size = 512

# Redis Configuration

redis\_node\_type = "cache.t3.micro"

# Domain Configuration

domain\_name = "staging.myproject.com"

# Monitoring

notification\_email = "dev-alerts@myproject.com"

# Feature Flags

enable\_monitoring = true

enable\_backup = false

```

\*\*Terraform Workflow Scripts:\*\*

```bash

#!/bin/bash

# scripts/terraform-deploy.sh

set -e

ENVIRONMENT=${1:-staging}

ACTION=${2:-plan}

if [[ ! -d "environments/$ENVIRONMENT" ]]; then

echo "Environment $ENVIRONMENT not found"

exit 1

fi

echo "🏗️ Running Terraform $ACTION for $ENVIRONMENT environment"

# Initialize Terraform

terraform init -reconfigure \

-backend-config="key=infrastructure/$ENVIRONMENT/terraform.tfstate"

# Select workspace

terraform workspace select $ENVIRONMENT || terraform workspace new $ENVIRONMENT

# Validate configuration

terraform validate

# Plan or Apply

case $ACTION in

"plan")

terraform plan \

-var-file="environments/$ENVIRONMENT/terraform.tfvars" \

-out="tfplan-$ENVIRONMENT"

;;

"apply")

terraform apply \

-var-file="environments/$ENVIRONMENT/terraform.tfvars" \

-auto-approve

;;

"destroy")

terraform destroy \

-var-file="environments/$ENVIRONMENT/terraform.tfvars" \

-auto-approve

;;

\*)

echo "Unknown action: $ACTION"

exit 1

;;

esac

echo "✅ Terraform $ACTION completed successfully!"

```

\*\*State Management & Security:\*\*

```hcl

# terraform-backend/main.tf

resource "aws\_s3\_bucket" "terraform\_state" {

bucket = "terraform-state-${random\_string.bucket\_suffix.result}"

tags = {

Name = "Terraform State Bucket"

Environment = "shared"

}

}

resource "aws\_s3\_bucket\_versioning" "terraform\_state" {

bucket = aws\_s3\_bucket.terraform\_state.id

versioning\_configuration {

status = "Enabled"

}

}

resource "aws\_s3\_bucket\_encryption" "terraform\_state" {

bucket = aws\_s3\_bucket.terraform\_state.id

server\_side\_encryption\_configuration {

rule {

apply\_server\_side\_encryption\_by\_default {

sse\_algorithm = "AES256"

}

}

}

}

resource "aws\_s3\_bucket\_public\_access\_block" "terraform\_state" {

bucket = aws\_s3\_bucket.terraform\_state.id

block\_public\_acls = true

block\_public\_policy = true

ignore\_public\_acls = true

restrict\_public\_buckets = true

}

# DynamoDB table for state locking

resource "aws\_dynamodb\_table" "terraform\_state\_lock" {

name = "terraform-state-lock"

billing\_mode = "PAY\_PER\_REQUEST"

hash\_key = "LockID"

attribute {

name = "LockID"

type = "S"

}

tags = {

Name = "Terraform State Lock Table"

Environment = "shared"

}

}

resource "random\_string" "bucket\_suffix" {

length = 8

special = false

upper = false

}

```

\*\*Cost Optimization & Monitoring:\*\*

- Resource Tagging: [COST\_ALLOCATION/RESOURCE\_TRACKING/AUTOMATED\_BILLING]

- Right-sizing: [INSTANCE\_OPTIMIZATION/STORAGE\_OPTIMIZATION/PERFORMANCE\_MONITORING]

- Lifecycle Management: [S3\_LIFECYCLE\_POLICIES/EBS\_SNAPSHOT\_CLEANUP/LOG\_RETENTION]

- Reserved Instances: [CAPACITY\_PLANNING/COST\_OPTIMIZATION/USAGE\_FORECASTING]

Provide complete Infrastructure as Code implementation with modular Terraform configuration, environment management, security best practices, and cost optimization strategies.

```

---

## Code Quality & Maintenance Templates

### 19. Code Review & Quality Assurance

```

Act as a senior software engineer specializing in code quality for lovable.dev and windsurf. Create a comprehensive code review and quality assurance process for [PROJECT\_NAME] that:

\*\*Code Review Process:\*\*

- Review Workflow: [PULL\_REQUEST\_PROCESS/BRANCH\_PROTECTION/AUTOMATED\_CHECKS]

- Review Criteria: [CODE\_QUALITY/SECURITY/PERFORMANCE/MAINTAINABILITY]

- Review Tools: [GITHUB\_REVIEWS/SONARQUBE/CODECLIMATE/CUSTOM\_TOOLS]

- Review Metrics: [REVIEW\_TIME/DEFECT\_DETECTION/CODE\_COVERAGE]

- Reviewer Assignment: [AUTOMATIC\_ASSIGNMENT/ROUND\_ROBIN/EXPERTISE\_BASED]

\*\*Code Quality Standards:\*\*

```yaml

# .github/pull\_request\_template.md

## Pull Request Checklist

### Code Quality

- [ ] Code follows project style guidelines

- [ ] All functions have appropriate documentation

- [ ] Code is self-documenting with clear variable names

- [ ] No commented-out code or console.log statements

- [ ] Error handling is implemented where appropriate

### Testing

- [ ] New features have corresponding tests

- [ ] All tests pass locally

- [ ] Test coverage is maintained or improved

- [ ] Edge cases are covered in tests

- [ ] Integration tests updated if needed

### Security

- [ ] No sensitive information exposed in code

- [ ] Input validation implemented where needed

- [ ] Authentication/authorization properly implemented

- [ ] SQL injection and XSS vulnerabilities addressed

- [ ] Dependencies security scan passed

### Performance

- [ ] No obvious performance bottlenecks

- [ ] Database queries optimized

- [ ] Large data structures handled efficiently

- [ ] Unnecessary API calls eliminated

- [ ] Image and asset optimization implemented

### Documentation

- [ ] README updated if needed

- [ ] API documentation updated

- [ ] Configuration changes documented

- [ ] Migration scripts documented

- [ ] Breaking changes noted

### Deployment

- [ ] Environment variables configured

- [ ] Database migrations work correctly

- [ ] Deployment scripts tested

- [ ] Rollback plan considered

- [ ] Monitoring and alerting updated

## Description

Brief description of changes and motivation.

## Testing

Describe testing approach and any manual testing performed.

## Screenshots (if applicable)

Include before/after screenshots for UI changes.

## Additional Notes

Any additional context or considerations for reviewers.

```

\*\*Automated Code Quality Checks:\*\*

```yaml

# .github/workflows/code-quality.yml

name: Code Quality Checks

on:

pull\_request:

branches: [main, develop]

jobs:

code-quality:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

with:

fetch-depth: 0 # Full history for better analysis

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: '18'

cache: 'npm'

- name: Install dependencies

run: npm ci

# ESLint for JavaScript/TypeScript

- name: Run ESLint

run: npx eslint . --ext .js,.jsx,.ts,.tsx --format json --output-file eslint-report.json

continue-on-error: true

- name: Annotate ESLint results

uses: ataylorme/eslint-annotate-action@v2

if: always()

with:

repo-token: "${{ secrets.GITHUB\_TOKEN }}"

report-json: "eslint-report.json"

# Prettier for code formatting

- name: Check code formatting

run: npx prettier --check .

# TypeScript type checking

- name: TypeScript type check

run: npx tsc --noEmit

if: hashFiles('tsconfig.json') != ''

# Unit tests with coverage

- name: Run tests with coverage

run: npm run test:coverage

- name: Upload coverage to Codecov

uses: codecov/codecov-action@v3

with:

file: ./coverage/lcov.info

flags: unittests

name: codecov-umbrella

# Security audit

- name: Run security audit

run: npm audit --audit-level=moderate

# Dependency check

- name: Check for outdated dependencies

run: npm outdated || true

# Bundle size analysis

- name: Analyze bundle size

run: |

npm run build

npx bundlesize

if: contains(github.event.pull\_request.changed\_files, 'package.json')

# SonarQube analysis

- name: SonarQube Scan

uses: sonarqube-quality-gate-action@master

env:

SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }}

# Lovable.dev specific checks

frontend-quality:

runs-on: ubuntu-latest

if: contains(github.event.pull\_request.changed\_files, 'frontend/')

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: '18'

- name: Install dependencies

working-directory: ./frontend

run: npm ci

- name: Accessibility tests

working-directory: ./frontend

run: npm run test:a11y

- name: Visual regression tests

working-directory: ./frontend

run: npm run test:visual

- name: Performance audit

working-directory: ./frontend

run: |

npm run build

npx lighthouse-ci autorun

# Windsurf specific checks

backend-quality:

runs-on: ubuntu-latest

if: contains(github.event.pull\_request.changed\_files, 'backend/')

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Node.js

uses: actions/setup-node@v4

with:

node-version: '18'

- name: Install dependencies

working-directory: ./backend

run: npm ci

- name: API contract testing

working-directory: ./backend

run: npm run test:contract

- name: Database migration tests

working-directory: ./backend

run: npm run test:migration

- name: Load testing

working-directory: ./backend

run: npm run test:load

```

\*\*Code Quality Configuration Files:\*\*

```json

// .eslintrc.json

{

"root": true,

"env": {

"browser": true,

"es2021": true,

"node": true,

"jest": true

},

"extends": [

"eslint:recommended",

"@typescript-eslint/recommended",

"react-hooks/recommended",

"prettier"

],

"parser": "@typescript-eslint/parser",

"parserOptions": {

"ecmaFeatures": {

"jsx": true

},

"ecmaVersion": 12,

"sourceType": "module",

"project": "./tsconfig.json"

},

"plugins": [

"react",

"@typescript-eslint",

"react-hooks",

"import",

"security",

"sonarjs"

],

"rules": {

// TypeScript specific rules

"@typescript-eslint/no-unused-vars": "error",

"@typescript-eslint/explicit-function-return-type": "warn",

"@typescript-eslint/no-explicit-any": "warn",

"@typescript-eslint/prefer-const": "error",

// React specific rules

"react/prop-types": "off",

"react/react-in-jsx-scope": "off",

"react-hooks/rules-of-hooks": "error",

"react-hooks/exhaustive-deps": "warn",

// Import rules

"import/order": ["error", {

"groups": ["builtin", "external", "internal", "parent", "sibling"],

"newlines-between": "always"

}],

"import/no-unused-modules": "warn",

// Security rules

"security/detect-object-injection": "warn",

"security/detect-sql-injection": "error",

"security/detect-xss-regex": "error",

// SonarJS rules for code quality

"sonarjs/cognitive-complexity": ["error", 15],

"sonarjs/max-switch-cases": ["error", 30],

"sonarjs/no-duplicate-string": ["error", 3],

"sonarjs/no-identical-functions": "error",

// General code quality

"complexity": ["error", 10],

"max-depth": ["error", 4],

"max-lines": ["error", 300],

"max-lines-per-function": ["error", 50],

"no-console": "warn",

"no-debugger": "error",

"prefer-const": "error"

},

"overrides": [

{

"files": ["\*\*/\*.test.ts", "\*\*/\*.test.tsx"],

"rules": {

"max-lines-per-function": "off",

"sonarjs/no-duplicate-string": "off"

}

}

]

}

```

```json

// .prettierrc

{

"semi": true,

"trailingComma": "es5",

"singleQuote": true,

"printWidth": 100,

"tabWidth": 2,

"useTabs": false,

"bracketSpacing": true,

"arrowParens": "avoid",

"endOfLine": "lf",

"overrides": [

{

"files": "\*.json",

"options": {

"tabWidth": 2

}

}

]

}

```

\*\*SonarQube Configuration:\*\*

```properties

# sonar-project.properties

sonar.projectKey=my-project

sonar.projectName=My Project

sonar.projectVersion=1.0

# Source code

sonar.sources=src

sonar.tests=src

sonar.test.inclusions=\*\*/\*.test.ts,\*\*/\*.test.tsx,\*\*/\*.spec.ts,\*\*/\*.spec.tsx

sonar.exclusions=\*\*/node\_modules/\*\*,\*\*/build/\*\*,\*\*/dist/\*\*,\*\*/\*.config.js

# Language specific settings

sonar.typescript.lcov.reportPaths=coverage/lcov.info

sonar.javascript.lcov.reportPaths=coverage/lcov.info

# Quality gate settings

sonar.qualitygate.wait=true

# Duplications

sonar.cpd.exclusions=\*\*/\*.test.ts,\*\*/\*.test.tsx

# Security

sonar.security.hotspots.inheritance=true

```

\*\*Code Review Guidelines:\*\*

```markdown

# Code Review Guidelines

## Review Priorities (in order)

### 1. Correctness

- Does the code do what it's supposed to do?

- Are there any logical errors or bugs?

- Are edge cases handled properly?

- Is error handling comprehensive?

### 2. Security

- Are there any security vulnerabilities?

- Is input validation implemented?

- Are authentication/authorization checks in place?

- Is sensitive data properly protected?

### 3. Performance

- Are there any obvious performance issues?

- Are database queries optimized?

- Is caching implemented where appropriate?

- Are there any memory leaks?

### 4. Maintainability

- Is the code readable and understandable?

- Are functions and variables named clearly?

- Is the code properly structured?

- Is there appropriate documentation?

### 5. Design

- Does the code follow SOLID principles?

- Is the architecture appropriate?

- Are there any code smells?

- Is the code DRY (Don't Repeat Yourself)?

## Review Process

### Before Starting Review

1. Understand the context and requirements

2. Check if all automated checks pass

3. Review the PR description and linked issues

4. Understand the scope of changes

### During Review

1. Focus on the most critical aspects first

2. Provide constructive feedback

3. Suggest specific improvements

4. Ask questions for clarification

5. Acknowledge good practices

### Review Comments Guidelines

- Be respectful and constructive

- Explain the "why" behind suggestions

- Provide examples when possible

- Use "nit:" for minor suggestions

- Distinguish between blocking and non-blocking issues

### Examples of Good Review Comments

\*\*Good:\*\*

```

Consider using a more descriptive variable name here. Instead of `data`,

perhaps `userProfiles` would make the code more self-documenting.

```

\*\*Better:\*\*

```

The variable name `data` is generic. Consider `userProfiles` to make

the code more readable:

```typescript

// Instead of

const data = await fetchUsers();

// Consider

const userProfiles = await fetchUsers();

```

\*\*Best:\*\*

```

Security concern: This endpoint doesn't validate user permissions.

Users could potentially access other users' data.

Consider adding an authorization check:

```typescript

if (req.user.id !== userId && !req.user.roles.includes('admin')) {

throw new ForbiddenError('Access denied');

}

```

Ref: OWASP A01:2021 - Broken Access Control

```

## Lovable.dev Specific Guidelines

### Frontend Code Review Focus

- Accessibility compliance (WCAG guidelines)

- Responsive design implementation

- Performance optimization (Core Web Vitals)

- SEO considerations

- Brand consistency

- Cross-browser compatibility

### Component Review Checklist

- [ ] Props are properly typed

- [ ] Component is accessible

- [ ] Responsive design works on all breakpoints

- [ ] Loading and error states handled

- [ ] Performance optimized (memoization, lazy loading)

- [ ] Proper semantic HTML used

## Windsurf Specific Guidelines

### Backend Code Review Focus

- API design and REST principles

- Database query optimization

- Security vulnerabilities

- Error handling and logging

- Performance and scalability

- AWS service integration

### API Review Checklist

- [ ] Input validation implemented

- [ ] Authentication/authorization checks

- [ ] Proper HTTP status codes used

- [ ] Error responses consistent

- [ ] Database queries optimized

- [ ] Caching implemented where appropriate

- [ ] Rate limiting considered

## Code Quality Metrics

### Acceptance Criteria

- Code coverage: minimum 80%

- Cognitive complexity: maximum 15

- Function length: maximum 50 lines

- File length: maximum 300 lines

- Dependency freshness: no critical vulnerabilities

### Automated Checks Must Pass

- All tests passing

- Linting with no errors

- Type checking (if TypeScript)

- Security audit passing

- Build successful

- Performance budget met

```

\*\*Review Automation Tools:\*\*

```javascript

// scripts/review-automation.js

const { Octokit } = require('@octokit/rest');

class ReviewAutomation {

constructor(token) {

this.octokit = new Octokit({ auth: token });

}

async autoAssignReviewers(owner, repo, pullNumber) {

// Get PR details

const { data: pr } = await this.octokit.pulls.get({

owner,

repo,

pull\_number: pullNumber,

});

// Determine reviewers based on changed files

const reviewers = this.getReviewersForFiles(pr.changed\_files);

// Assign reviewers

if (reviewers.length > 0) {

await this.octokit.pulls.requestReviewers({

owner,

repo,

pull\_number: pullNumber,

reviewers,

});

}

}

getReviewersForFiles(files) {

const reviewerMapping = {

'frontend/': ['frontend-team-lead', 'ui-specialist'],

'backend/': ['backend-team-lead', 'api-specialist'],

'infrastructure/': ['devops-lead', 'infrastructure-specialist'],

'docs/': ['tech-writer', 'product-manager'],

};

const reviewers = new Set();

files.forEach(file => {

Object.entries(reviewerMapping).forEach(([path, teamReviewers]) => {

if (file.filename.startsWith(path)) {

teamReviewers.forEach(reviewer => reviewers.add(reviewer));

}

});

});

return Array.from(reviewers).slice(0, 2); // Limit to 2 reviewers

}

async addQualityLabels(owner, repo, pullNumber) {

const { data: pr } = await this.octokit.pulls.get({

owner,

repo,

pull\_number: pullNumber,

});

const labels = [];

// Add size labels

if (pr.additions + pr.deletions > 500) {

labels.push('size/large');

} else if (pr.additions + pr.deletions > 100) {

labels.push('size/medium');

} else {

labels.push('size/small');

}

// Add type labels based on files

const hasTests = pr.changed\_files.some(file =>

file.filename.includes('.test.') || file.filename.includes('.spec.')

);

if (!hasTests && pr.additions > 50) {

labels.push('needs-tests');

}

// Add labels

if (labels.length > 0) {

await this.octokit.issues.addLabels({

owner,

repo,

issue\_number: pullNumber,

labels,

});

}

}

}

module.exports = ReviewAutomation;

```

\*\*Quality Metrics Dashboard:\*\*

```javascript

// scripts/quality-metrics.js

const fs = require('fs');

const path = require('path');

class QualityMetrics {

constructor() {

this.metrics = {

codeQuality: {},

testCoverage: {},

security: {},

performance: {},

maintainability: {}

};

}

async generateReport() {

// Collect metrics from various sources

this.metrics.codeQuality = await this.getCodeQualityMetrics();

this.metrics.testCoverage = await this.getTestCoverageMetrics();

this.metrics.security = await this.getSecurityMetrics();

this.metrics.performance = await this.getPerformanceMetrics();

this.metrics.maintainability = await this.getMaintainabilityMetrics();

// Generate HTML report

const report = this.generateHTMLReport();

// Save report

fs.writeFileSync('quality-report.html', report);

console.log('Quality metrics report generated: quality-report.html');

}

async getCodeQualityMetrics() {

// ESLint results

const eslintReport = JSON.parse(

fs.readFileSync('eslint-report.json', 'utf8')

);

return {

totalIssues: eslintReport.reduce((sum, file) => sum + file.errorCount + file.warningCount, 0),

errors: eslintReport.reduce((sum, file) => sum + file.errorCount, 0),

warnings: eslintReport.reduce((sum, file) => sum + file.warningCount, 0),

};

}

async getTestCoverageMetrics() {

// Jest coverage report

const coverageReport = JSON.parse(

fs.readFileSync('coverage/coverage-summary.json', 'utf8')

);

return {

lines: coverageReport.total.lines.pct,

statements: coverageReport.total.statements.pct,

functions: coverageReport.total.functions.pct,

branches: coverageReport.total.branches.pct,

};

}

generateHTMLReport() {

return `

<!DOCTYPE html>

<html>

<head>

<title>Code Quality Report</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

.metric { background: #f5f5f5; padding: 10px; margin: 10px 0; border-radius: 5px; }

.good { background: #d4edda; border-left: 4px solid #28a745; }

.warning { background: #fff3cd; border-left: 4px solid #ffc107; }

.danger { background: #f8d7da; border-left: 4px solid #dc3545; }

</style>

</head>

<body>

<h1>Code Quality Report</h1>

<p>Generated on: ${new Date().toISOString()}</p>

<div class="metric ${this.getQualityClass(this.metrics.testCoverage.lines)}">

<h3>Test Coverage</h3>

<p>Lines: ${this.metrics.testCoverage.lines}%</p>

<p>Functions: ${this.metrics.testCoverage.functions}%</p>

<p>Branches: ${this.metrics.testCoverage.branches}%</p>

</div>

<div class="metric ${this.getQualityClass(this.metrics.codeQuality.totalIssues, true)}">

<h3>Code Quality</h3>

<p>Total Issues: ${this.metrics.codeQuality.totalIssues}</p>

<p>Errors: ${this.metrics.codeQuality.errors}</p>

<p>Warnings: ${this.metrics.codeQuality.warnings}</p>

</div>

</body>

</html>

`;

}

getQualityClass(value, inverse = false) {

if (inverse) {

return value === 0 ? 'good' : value < 5 ? 'warning' : 'danger';

}

return value >= 80 ? 'good' : value >= 60 ? 'warning' : 'danger';

}

}

module.exports = QualityMetrics;

```

\*\*Integration with AWS:\*\*

- CodeGuru Reviewer: [AUTOMATED\_CODE\_REVIEW/BEST\_PRACTICE\_DETECTION]

- CodeCommit: [REPOSITORY\_MANAGEMENT/BRANCH\_PROTECTION/MERGE\_POLICIES]

- CloudWatch: [QUALITY\_METRICS\_MONITORING/ALERTS/DASHBOARDS]

- Lambda: [AUTOMATED\_QUALITY\_CHECKS/REPORT\_GENERATION]

Provide complete code review and quality assurance system with automated checks, review guidelines, quality metrics, and integration with development workflow.

```

---

## Collaboration & Project Management Templates

### 20. Agile Development & Team Coordination

```

Act as an Agile coach and project manager specializing in software development teams using lovable.dev and windsurf. Create a comprehensive Agile development process for [PROJECT\_NAME] that:

\*\*Agile Framework Implementation:\*\*

- Methodology: [SCRUM/KANBAN/SCRUMBAN/CUSTOM\_HYBRID]

- Sprint Duration: [1\_WEEK/2\_WEEKS/3\_WEEKS/VARIABLE\_LENGTH]

- Team Structure: [CROSS\_FUNCTIONAL/SPECIALIZED/HYBRID\_APPROACH]

- Roles Definition: [PRODUCT\_OWNE