# Prompt Examples for AI Code Generation

Based on Software Requirements

This document provides prompt examples that help AI generate valid code from software requirements specifications. These prompts are designed to translate business requirements into technical implementations.

## 1. Introduction Section Prompts

### 1.1 Purpose Translation Prompt

Based on this software purpose: "This document describes what we want our online store website to do - let customers browse products, add them to a shopping cart, and complete purchases securely."   
  
Generate a technical project overview that includes:

1. The main technology stack recommendation (e.g., React/Vue for frontend, Node.js/Spring Boot for backend, PostgreSQL for database)
2. Core architectural patterns to implement (e.g., MVC, Microservices, etc.)
3. Security considerations for handling purchases
4. Key components that will be needed

### 1.2 Scope Translation Prompt

Based on this software scope: "Our online store will be used by customers shopping from home or work. They'll be able to shop 24/7 from any device with internet, and we'll save time by not needing a physical store."   
  
Generate a technical requirements specification that includes:

1. Supported platforms and devices
2. User authentication approach
3. Expected concurrent user load
4. Required uptime and availability targets
5. Cross-platform compatibility requirements

## 2. Overall Description Prompts

### 2.1 Product Perspective Translation Prompt

Based on this product perspective: "Our customer website will be separate from our internal systems for now. Customers can use it without needing to log into anything else, but eventually we'd like to connect it to our inventory system."   
  
Generate a system architecture plan that includes:

1. Initial system boundaries and components
2. API design for future inventory system integration
3. Data synchronization strategy between systems
4. Security measures for system integration
5. Microservices vs monolith recommendation

### 2.2 Product Functions Translation Prompt

Based on these product functions:

1. Show a catalog of all our products
2. Let customers save items to a shopping cart
3. Process payments securely
4. Send order confirmations by email

Generate a feature implementation plan with:

1. Database schema design for products and shopping cart
2. API endpoints for each function
3. Frontend component structure
4. Payment processing integration approach
5. Email notification system design

## 3. Specific Requirements Prompts

### 3.1 Functional Requirements Translation Prompts

#### User Accounts Implementation Prompt

Based on these user account requirements:

* People can create accounts with email and password
* People can log in to their accounts
* People can reset their password if they forget it
* The system sends a confirmation email when they sign up

Generate a complete user authentication system with:

1. User model/schema with validation
2. Password hashing and security implementation
3. Email verification workflow
4. Password reset functionality with secure tokens
5. Session management
6. REST API endpoints for all authentication operations
7. Frontend forms for registration, login, and password reset

Use [SPECIFY TECHNOLOGY STACK] and follow security best practices.

#### Product Catalog Implementation Prompt

Based on these product catalog requirements:

* Show all products on categorized pages
* Let customers search for products by name or description
* Let customers filter products by category, price, or features
* Show clear photos and descriptions for each product
* Show prices clearly, including any discounts

Generate a product catalog system with:

1. Product model/schema with categories, pricing, and media
2. Database indexing strategy for search and filtering
3. API endpoints for product listing with pagination
4. Search functionality implementation
5. Filtering mechanism by multiple criteria
6. Frontend components for product display and filtering
7. Image optimization and responsive display

Use [SPECIFY TECHNOLOGY STACK] and optimize for performance.

#### Shopping Cart Implementation Prompt

Based on these shopping cart requirements:

* Add items to a virtual shopping cart
* See what's in the cart at any time
* Change quantities or remove items
* See the total cost including tax and shipping
* Save the cart to finish shopping later (for logged-in users)

Generate a shopping cart system with:

1. Cart model/schema for session and persistent storage
2. Cart operations: add, update, remove items
3. Tax and shipping calculation logic
4. Real-time total calculation
5. API endpoints for cart operations
6. Frontend cart UI components
7. Persistence strategy for logged-in users

Use [SPECIFY TECHNOLOGY STACK] and ensure data consistency.

### 3.2 Non-Functional Requirements Translation Prompts

#### Performance Requirements Implementation Prompt

Based on these performance requirements:

* The website should load quickly (under 3 seconds)
* Searching for products should be instant (under 1 second)
* The system should handle many customers shopping at once

Generate a performance optimization plan that includes:

1. Frontend optimization techniques (lazy loading, code splitting, caching)
2. Backend optimization strategies (database indexing, query optimization)
3. CDN implementation for static assets
4. Caching strategy for product data and search results
5. Load testing approach to validate performance targets
6. Monitoring and alerting for performance metrics

Use [SPECIFY TECHNOLOGY STACK] and follow performance best practices.

#### Security Requirements Implementation Prompt

Based on these security requirements:

* Customer passwords should be kept secret and secure
* Customer information should be backed up regularly
* Only authorized people should access order information
* All communication should be encrypted

Generate a security implementation plan that includes:

1. Password hashing and storage approach
2. Data backup and recovery strategy
3. Role-based access control for order information
4. HTTPS/SSL implementation
5. Input validation and sanitization
6. Security headers and CORS configuration
7. Vulnerability scanning and testing approach

Use [SPECIFY TECHNOLOGY STACK] and follow security best practices.

#### Usability Requirements Implementation Prompt

Based on these usability requirements:

* First-time visitors should understand how to shop within 2 minutes
* The website should be easy to navigate with clear buttons
* Error messages should explain what went wrong in plain language
* The website should work on phones, tablets, and computers

Generate a usability implementation plan that includes:

1. User interface design principles and component library
2. Responsive design implementation approach
3. Intuitive navigation structure
4. Clear error messaging strategy
5. User onboarding and guidance features
6. Accessibility compliance (WCAG)
7. User testing approach

Use [SPECIFY TECHNOLOGY STACK] and follow UX best practices.

## 4. External Interface Requirements Prompts

### User Interface Design Prompt

Based on this UI requirement: "The website should look clean and professional. Customers should see our logo at the top, easy navigation to product categories across the top, a search bar, and their shopping cart in the upper right corner."   
  
Generate a frontend implementation plan that includes:

1. Component hierarchy and structure
2. Styling approach (CSS framework or custom)
3. Responsive navigation bar implementation
4. Header component with logo, navigation, search, and cart
5. Design system with colors, typography, and spacing
6. Mobile-friendly navigation approach

Use [SPECIFY FRONTEND TECHNOLOGY] and follow modern UI/UX principles.

### Software Integration Prompt

Based on this integration requirement: "The website will connect to a payment company to process credit cards and PayPal payments."   
  
Generate a payment processing integration plan that includes:

1. Payment service provider selection and integration approach
2. API integration for credit card and PayPal processing
3. Security measures for payment data handling
4. Error handling and fallback mechanisms
5. Testing strategy for payment flows
6. Compliance requirements (PCI DSS)

Use [SPECIFY TECHNOLOGY STACK] and follow payment processing best practices.

## 5. Other Requirements Prompts

### Legal Compliance Implementation Prompt

Based on this legal requirement: "Must follow privacy laws to protect customer information"   
  
Generate a privacy compliance implementation plan that includes:

1. Data protection measures (GDPR, CCPA, etc.)
2. Privacy policy implementation
3. Cookie consent mechanism
4. Data retention and deletion policies
5. User data access and portability features
6. Audit logging for compliance tracking

Use [SPECIFY TECHNOLOGY STACK] and follow privacy compliance best practices.

## 6. Complete Feature Implementation Prompts

### End-to-End Feature Implementation Prompt

Based on these requirements for a shopping cart feature:   
  
Description: Customers need to save items while they shop and checkout when ready.   
  
Features:

* Add items to a virtual shopping cart
* See what's in the cart at any time
* Change quantities or remove items
* See the total cost including tax and shipping
* Save the cart to finish shopping later (for logged-in users)

Generate a complete implementation including:

1. Database schema design
2. Backend API endpoints with documentation
3. Frontend components with state management
4. Business logic for cart operations
5. Tax and shipping calculation logic
6. User authentication integration
7. Error handling and validation
8. Testing strategy with unit and integration tests

Use [SPECIFY FULL TECHNOLOGY STACK] and follow best practices for each technology.

## 7. Best Practices for Using These Prompts

### Prompt Customization Guidelines

1. **Replace placeholders**: Always replace [SPECIFY TECHNOLOGY STACK] with actual technologies
2. **Add specificity**: Include version numbers, hosting requirements, or specific libraries when known
3. **Adjust scope**: Modify the requirements based on project complexity and timeline
4. **Include constraints**: Add any technical constraints, budget limitations, or team expertise considerations

### Iterative Development Prompts

Based on the previously generated [FEATURE NAME] implementation:

1. Identify the most critical 20% of functionality for MVP
2. Create a phased development plan with clear milestones
3. Specify which components can be developed in parallel
4. Identify potential technical risks and mitigation strategies

### Testing and Quality Assurance Prompt

Based on the implemented [FEATURE NAME]:

1. Generate unit tests for all business logic
2. Create integration tests for API endpoints
3. Develop end-to-end tests for user workflows
4. Define performance benchmarks and load tests
5. Create security testing plan
6. Establish code quality standards and linting rules

## 8. Template for Creating New Prompts

When creating new prompts for requirements not covered above, use this template:

Based on this requirement: "[EXACT REQUIREMENT TEXT]"   
  
Generate [SPECIFIC DELIVERABLE] that includes:

1. [TECHNICAL COMPONENT 1]
2. [TECHNICAL COMPONENT 2]
3. [TECHNICAL COMPONENT 3]
4. [ADDITIONAL CONSIDERATIONS]

Use [TECHNOLOGY STACK] and follow [RELEVANT BEST PRACTICES].

This approach ensures that business requirements are accurately translated into technical implementations while maintaining consistency and quality.

Prompt Examples for AI Code Generation Based on Software Requirements