Development and Deployment Guide

Table of Contents

System Requirements

- Node.js (latest version)
- Bun Runtime
- Git
- VS Code (recommended)

Required VS Code Extensions

- Tailwind CSS IntelliSense
- ES7+ React/Redux/React-Native snippets
- Biome Extension

Installation and Setup

1. Clone the project:

```
git clone [repository-url]
cd react-vite-with-unit-test
```

2. Install dependencies:

```
bun install
```

3. Environment setup:
   - Copy `.env.example` to `.env.development`
   - Update necessary environment variables
4. Set up Git hooks:

```
bun prepare
```

Development Process

1. Start Development Environment

```
bun dev
```

2. Component Development

1. Create new component:

```
bun generate
```

2. Develop in Storybook:

```
bun storybook
```

3. Coding Workflow

1. Create new branch:

```
git checkout -b feature/feature-name
```

2. Format code before committing:

```
bun format
bun lint
```

3. Check code:

```
bun check
```

Testing

1. Unit Tests

```
# Run all tests
bun test

# Run tests in watch mode
bun test:watch

# Check coverage
bun test:coverage
```

2. Storybook Tests

```
# Run Storybook
bun storybook

# Build Storybook
bun build-storybook
```

Deployment

1. Production Build

```
bun build
```

2. Preview Build

```
bun preview
```

3. Docker Deployment

```
# Build Docker image
docker build -t react-vite-app .

# Run container
docker run -p 8080:80 react-vite-app
```

Rules and Standards

1. Code Style

- Use Biome for formatting and linting
- Follow TypeScript strict mode
- Use functional components and hooks

2. Git Commit

- Use conventional commits
- Each commit must pass all tests
- Pre-commit hooks will check:
  - Code formatting
  - Linting
  - Type checking
  - Unit tests

3. Component Development

1. Directory Structure:

```
src/
  ├── components/
  │   ├── [ComponentName]/
  │   │   ├── index.tsx
  │   │   ├── [ComponentName].stories.tsx
  │   │   └── [ComponentName].test.tsx
  │   └── ...
  ├── hooks/
  ├── utils/
  └── ...
```

2. Component Guidelines:

- Use TypeScript interfaces for props
- Implement error boundaries
- Optimize performance with React.memo when necessary
- Write stories for all cases

4. Testing Guidelines

- Minimum coverage: 80%
- Test edge cases
- Mock external dependencies
- Follow testing-library best practices

5. Performance Optimization

- Lazy loading for routes
- Code splitting
- Image optimization
- PWA implementation

CI/CD Pipeline

GitHub Actions workflow includes:

1. Build check

2. Unit tests
3. Storybook build
4. Docker image build
5. Deployment to staging/production

Monitoring and Logging

1. Error tracking with Error Boundary
2. Performance monitoring
3. User analytics

Security Guidelines

1. Dependency scanning
2. Regular updates
3. Security best practices
4. Environment variables management

Support

If you encounter issues:

1. Check documentation
2. Create an issue on GitHub
3. Contact team lead