Development Guidelines

AI-Powered Personal Productivity System

4.1 Code Standards

TypeScript Configuration

json

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "lib": ["ES2022", "DOM", "DOM.Iterable", "WebWorker"],
    "jsx": "react-jsx",
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "allowSyntheticDefaultImports": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "types": ["vite/client", "node"]
  }
}
```

Component Structure

typescript

```typescript
// Required structure for all components

interface ComponentSpec {
  // Props interface with JSDoc
  props: {
    required: Record<string, Type>;
    optional?: Record<string, Type>;
  };

  // State interface if stateful
  state?: Record<string, Type>;

  // Public methods interface
  methods?: Record<string, Function>;

  // Events emitted
  events?: Record<string, EventType>;

  // Performance budgets
  performance: {
    renderTime: number; // ms
    memoryLimit: number; // MB
    updateFrequency: number; // per second
  };

  // Accessibility requirements
  a11y: {
    role: string;
    ariaLabel: boolean;
    keyboardNav: boolean;
    screenReader: boolean;
  };
}
```

Error Handling Standards

typescript

```
// Error classification and handling

enum ErrorSeverity {
  CRITICAL = 'critical',  // System crash, data loss risk
  HIGH = 'high',          // Feature broken, needs immediate fix
  MEDIUM = 'medium',      // Degraded experience
  LOW = 'low'             // Minor issue, can be ignored
}

interface ErrorHandler {
  log(error: Error, severity: ErrorSeverity, context?: any): void;
  recover(error: Error): boolean;
  report(error: Error): Promise<void>;
  display(error: Error, userMessage: string): void;
}

// Required error boundaries for each major component
class TaskErrorBoundary extends ErrorBoundary {
  static fallback = TaskErrorFallback;
  static resetKeys = ['userId', 'taskId'];
  static resetOnPropsChange = true;
}
```

Naming Conventions

typescript

```
// File naming
ComponentName.tsx      // React components
useHookName.ts         // Custom hooks
serviceName.ts         // Service classes
utilityName.ts         // Utility functions
ComponentName.test.tsx // Test files
ComponentName.module.css // CSS modules

// Variable naming
const MAX_RETRY_COUNT = 3;       // Constants in UPPER_SNAKE_CASE
const userId = 'uuid';           // Variables in camelCase
const TaskManager = {};          // Classes/Components in PascalCase
const handleTaskCreate = () => {}; // Functions in camelCase

// Interface/Type naming
interface TaskProps {}           // Interface with Props suffix
type TaskState = {};             // Type with descriptive name
enum TaskStatus {}               // Enum in PascalCase
```

Code Organization

typescript

```tsx
// Import order
import React from 'react';                    // 1. React
import { useState, useEffect } from 'react';  // 2. React hooks
import { useRouter } from 'next/router';      // 3. External libraries
import { TaskCard } from '@/components';      // 4. Internal components
import { useAuth } from '@/hooks';            // 5. Internal hooks
import { taskService } from '@/services';     // 6. Services
import { formatDate } from '@/utils';         // 7. Utilities
import type { Task } from '@/types';          // 8. Types
import styles from './Component.module.css';  // 9. Styles

// Component structure
export const Component: React.FC<Props> = ({ prop1, prop2 }) => {
  // 1. Hooks
  const [state, setState] = useState();

  // 2. Derived state
  const derivedValue = useMemo(() => {}, []);

  // 3. Effects
  useEffect(() => {}, []);

  // 4. Handlers
  const handleClick = useCallback(() => {}, []);

  // 5. Render helpers
  const renderItem = () => {};

  // 6. Main render
  return <div>{content}</div>;
};
```

4.2 Testing Requirements

Test Coverage Targets

Component Type	Unit Tests	Integration	E2E	Total
Core Logic	95%	85%	70%	90%
UI Components	85%	75%	60%	80%
API Endpoints	90%	90%	80%	90%
AI Functions	85%	80%	70%	85%

Component Type	Unit Tests	Integration	E2E	Total
Sync Engine	95%	90%	85%	92%

Test Specifications

typescript

```typescript
// Test structure requirements

describe('Component/Feature Name', () => {
  // Setup and teardown
  beforeAll(() => { /* Global setup */ });
  beforeEach(() => { /* Test setup */ });
  afterEach(() => { /* Test cleanup */ });
  afterAll(() => { /* Global cleanup */ });

  // Functional tests
  describe('Functionality', () => {
    it('should handle normal input', () => {});
    it('should validate edge cases', () => {});
    it('should handle errors gracefully', () => {});
  });

  // Performance tests
  describe('Performance', () => {
    it('should render within 16ms', () => {});
    it('should handle 10k items', () => {});
  });

  // Accessibility tests
  describe('Accessibility', () => {
    it('should be keyboard navigable', () => {});
    it('should have proper ARIA labels', () => {});
    it('should support screen readers', () => {});
  });

  // Cross-platform tests
  describe('Compatibility', () => {
    it.each(['ios', 'android', 'desktop'])
      ('should work on %s', (platform) => {});
  });
});
```

Testing Utilities

```typescript
// Custom testing utilities
export const renderWithProviders = (component, options = {}) => {
  const AllTheProviders = ({ children }) => (
    <ThemeProvider>
      <AuthProvider>
        <QueryClient>
          {children}
        </QueryClient>
      </AuthProvider>
    </ThemeProvider>
  );

  return render(component, { wrapper: AllTheProviders, ...options });
};

// Mock data generators
export const createMockTask = (overrides = {}): Task => ({
  id: faker.datatype.uuid(),
  title: faker.lorem.sentence(),
  status: 'pending',
  priority: faker.datatype.number({ min: 0, max: 10 }),
  createdAt: faker.date.recent(),
  ...overrides
});

// Test helpers
export const waitForAsync = () =>
  new Promise(resolve => setTimeout(resolve, 0));

export const mockApiResponse = (data, delay = 0) =>
  new Promise(resolve => setTimeout(() => resolve(data), delay));
```

4.3 Performance Budgets

Critical Metrics

Metric	Target	Maximum	Measurement Tool
First Contentful Paint	1.0s	1.5s	Lighthouse
Time to Interactive	2.5s	3.5s	Lighthouse

Metric	Target	Maximum	Measurement Tool
Largest Contentful Paint	2.0s	2.5s	Web Vitals
First Input Delay	50ms	100ms	Web Vitals
Cumulative Layout Shift	0.05	0.1	Web Vitals
JavaScript Bundle Size	150KB	200KB	Webpack
CSS Bundle Size	30KB	50KB	Webpack
Memory Usage	100MB	200MB	Performance API
Task Operation	50ms	100ms	Custom metrics
Sync Operation	500ms	1000ms	Custom metrics

Resource Budgets

javascript

```javascript
// Bundle size limits per route
{
  "/": {
    js: 100,  // KB
    css: 20,
    total: 150
  },
  "/tasks": {
    js: 150,
    css: 30,
    total: 200
  },
  "/settings": {
    js: 50,
    css: 10,
    total: 75
  }
}

// API response size limits
{
  "tasks/list": 100,    // KB
  "tasks/single": 10,
  "sync/delta": 50,
  "export/full": 1000
}
```

Performance Optimization Strategies

javascript

```javascript
// Code splitting
const TaskEditor = lazy(() => import('./TaskEditor'));

// Image optimization
<Image
  src="/hero.jpg"
  alt="Hero"
  width={1200}
  height={600}
  loading="lazy"
  placeholder="blur"
/>

// Memoization
const expensiveCalculation = useMemo(() => {
  return calculateComplexValue(data);
}, [data]);

// Virtual scrolling for lists
import { VariableSizeList } from 'react-window';

<VariableSizeList
  height={600}
  itemCount={tasks.length}
  itemSize={getItemSize}
  width="100%"
>
  {TaskRow}
</VariableSizeList>

// Debouncing user input
const debouncedSearch = useMemo(
  () => debounce(handleSearch, 300),
  []
);
```

4.4 Documentation Standards

Code Documentation

typescript

```typescript
/**
 * TaskManager - Handles all task-related operations
 *
 * @class TaskManager
 * @implements {ITaskManager}
 *
 * @example
 * const manager = new TaskManager();
 * const task = await manager.createTask({ title: 'New Task' });
 */
export class TaskManager implements ITaskManager {
  /**
   * Creates a new task with the given properties
   *
   * @param {CreateTaskDTO} taskData - Task creation data
   * @param {TaskOptions} options - Additional options
   * @returns {Promise<Task>} The created task
   * @throws {ValidationError} If task data is invalid
   * @throws {DatabaseError} If database operation fails
   *
   * @example
   * const task = await createTask(
   *   { title: 'Review PR', priority: 8 },
   *   { notify: true }
   * );
   */
  async createTask(
    taskData: CreateTaskDTO,
    options: TaskOptions = {}
  ): Promise<Task> {
    // Implementation
  }
}
```

README Structure

markdown

```
# Component/Feature Name

## Overview
Brief description of what this component/feature does.

## Installation
```bash
npm install [dependencies]
```

Usage

```typescript
import { Component } from './Component';

<Component prop1="value" onEvent={handler} />
```

Props

Name	Type	Default	Required	Description
prop1	string	-	Yes	Description
prop2	number	0	No	Description

Events

Event	Payload	Description
onTaskCreate	Task	Fired when task is created

Examples

[Provide 2-3 practical examples]

API Reference

[Detailed API documentation]

Testing

```bash
npm test Component.test.tsx
```
```

Performance Considerations

- Virtual scrolling for lists > 100 items
- Lazy loading for images
- Memoization for expensive calculations

```
## 4.5 Git Workflow

### Branch Strategy

```bash
main          # Production-ready code
├── develop    # Integration branch
    ├── feature/task-editor    # Feature branches
    ├── feature/ai-integration
    ├── bugfix/sync-issue     # Bug fixes
    └── hotfix/critical-bug   # Emergency fixes
```

Commit Message Format

```
bash

# Format: <type>(<scope>): <subject>

feat(tasks): add bulk task operations
fix(sync): resolve conflict resolution bug
docs(api): update REST endpoint documentation
style(ui): improve task card spacing
refactor(auth): simplify token refresh logic
test(tasks): add edge case tests for parser
perf(list): implement virtual scrolling
chore(deps): update dependencies

# Commit message rules:
# - Use present tense ("add" not "added")
# - Keep subject line under 50 characters
# - Reference issue numbers when applicable
# - Include breaking changes in footer
```
```

Pull Request Template

markdown

```
## Description
Brief description of changes

## Type of Change
- [ ] Bug fix
- [ ] New feature
- [ ] Breaking change
- [ ] Documentation update

## Testing
- [ ] Unit tests pass
- [ ] Integration tests pass
- [ ] Manual testing completed

## Checklist
- [ ] Code follows style guidelines
- [ ] Self-review completed
- [ ] Documentation updated
- [ ] No new warnings
- [ ] Performance impact assessed

## Related Issues
Fixes #123
```

Document Version: 1.0.0

Last Updated: January 2024

Next Review: February 2024