Quality Assurance Specifications

AI-Powered Personal Productivity System

7.1 Test Scenarios

Critical User Journeys

```gherkin
Feature: Task Management
  Scenario: Create task from natural language
    Given I am on the task page
    When I type "Meeting with John tomorrow at 3pm #work"
    Then a task should be created with:
      | title | Meeting with John |
      | due | tomorrow 15:00 |
      | tag | work |
    And the task should sync to cloud
    And the task should appear on other devices

  Scenario: Offline task creation
    Given I am offline
    When I create 5 tasks
    And I go online
    Then all 5 tasks should sync
    And no duplicates should exist
    And the order should be preserved

  Scenario: Conflict resolution
    Given task "Review" exists on two devices
    When device A marks it complete
    And device B updates the title
    Then the task should be complete with new title
    And both devices should show the same state
```

End-to-End Test Suites

```typescript
```

```
// E2E Test Suite Structure

describe('Task Lifecycle', () => {
  beforeEach(() => {
    cy.visit('/');
    cy.login('test@example.com', 'password');
  });

  it('should complete full task lifecycle', () => {
    // Create
    cy.get('[data-testid="create-task"]').click();
    cy.get('[data-testid="task-input"]').type('Test task with deadline tomorrow');
    cy.get('[data-testid="save-task"]').click();

    // Verify
    cy.get('[data-testid="task-list"]')
      .should('contain', 'Test task with deadline')
      .and('contain', 'Tomorrow');

    // Edit
    cy.get('[data-testid="task-1"]').dblclick();
    cy.get('[data-testid="task-priority"]').select('high');
    cy.get('[data-testid="save-task"]').click();

    // Complete
    cy.get('[data-testid="task-checkbox-1"]').click();
    cy.get('[data-testid="task-1"]').should('have.class', 'completed');

    // Delete
    cy.get('[data-testid="task-menu-1"]').click();
    cy.get('[data-testid="delete-task"]').click();
    cy.get('[data-testid="confirm-delete"]').click();
    cy.get('[data-testid="task-1"]').should('not.exist');
  });

  it('should handle bulk operations', () => {
    // Create multiple tasks
    const tasks = ['Task 1', 'Task 2', 'Task 3'];
    tasks.forEach(task => {
      cy.createTask(task);
    });

    // Select all
```

```
    cy.get('[data-testid="select-all"]').click();

    // Bulk update
    cy.get('[data-testid="bulk-actions"]').click();
    cy.get('[data-testid="bulk-set-priority"]').click();
    cy.get('[data-testid="priority-high"]').click();

    // Verify
    cy.get('[data-testid^="task-"]').each(($task) => {
      cy.wrap($task).should('have.attr', 'data-priority', 'high');
    });
  });
});
```

Edge Cases & Error Scenarios

Scenario	Expected Behavior	Recovery
10,000 tasks loaded	<2s render, virtual scroll	Pagination
Network timeout during sync	Queue operation, retry	Exponential backoff
Corrupt local database	Detect on startup	Restore from cloud
Invalid Claude command	Show error, preserve state	Suggest correction
Storage quota exceeded	Warn user, offer cleanup	Archive old tasks
Circular task dependency	Prevent creation	Show dependency graph
Malformed recurring pattern	Fallback to simple repeat	Show pattern builder

Performance Test Scenarios

javascript

```
// Performance test suite
describe('Performance Tests', () => {
  it('should render 10,000 tasks efficiently', async () => {
    const tasks = generateMockTasks(10000);

    const startTime = performance.now();
    await renderTaskList(tasks);
    const renderTime = performance.now() - startTime;

    expect(renderTime).toBeLessThan(2000);
    expect(getMemoryUsage()).toBeLessThan(200 * 1024 * 1024); // 200MB
  });

  it('should handle rapid task creation', async () => {
    const operations = [];

    for (let i = 0; i < 100; i++) {
      operations.push(createTask(`Task ${i}`));
    }

    const startTime = performance.now();
    await Promise.all(operations);
    const totalTime = performance.now() - startTime;

    expect(totalTime / 100).toBeLessThan(50); // <50ms per task
  });

  it('should maintain 60fps during scroll', async () => {
    const frameTimings = [];

    const measureFrame = () => {
      frameTimings.push(performance.now());
      if (frameTimings.length < 60) {
        requestAnimationFrame(measureFrame);
      }
    };

    requestAnimationFrame(measureFrame);
    await simulateScroll(1000); // Scroll 1000px

    const frameDurations = frameTimings.slice(1).map((time, i) =>
      time - frameTimings[i]
    );
```

```
    const avg = frameDurations.reduce((a, b) => a + b) / frameDurations.length;
    expect(avg).toBeLessThan(16.67); // 60fps = 16.67ms per frame
  });
});
```

7.2 Accessibility Testing

WCAG 2.1 AA Compliance

yaml

```
Perceivable:
  ✓ Text contrast ratio ≥4.5:1
  ✓ Focus indicators visible
  ✓ Images have alt text
  ✓ Error messages clear
  ✓ Color not sole indicator

Operable:
  ✓ Keyboard navigation complete
  ✓ No keyboard traps
  ✓ Skip links present
  ✓ Touch targets ≥44x44px
  ✓ No seizure-inducing content

Understandable:
  ✓ Language declared
  ✓ Labels describe purpose
  ✓ Instructions clear
  ✓ Error recovery guided
  ✓ Consistent navigation

Robust:
  ✓ Valid HTML
  ✓ ARIA used correctly
  ✓ Works with screen readers
  ✓ Browser zoom to 200%
  ✓ Progressive enhancement
```

Automated Accessibility Tests

javascript

```
// Axe-core integration
describe('Accessibility Tests', () => {
  beforeEach(() => {
    cy.visit('/');
    cy.injectAxe();
  });

  it('should have no accessibility violations on load', () => {
    cy.checkA11y();
  });

  it('should maintain accessibility during interactions', () => {
    // Open modal
    cy.get('[data-testid="create-task"]').click();
    cy.checkA11y('[data-testid="task-modal"]');

    // Navigate with keyboard
    cy.get('body').tab();
    cy.focused().should('have.attr', 'data-testid', 'task-input');

    // Check form errors
    cy.get('[data-testid="save-task"]').click();
    cy.checkA11y('[data-testid="error-message"]');
  });

  it('should support screen readers', () => {
    // Check ARIA labels
    cy.get('[role="button"]').each(($button) => {
      cy.wrap($button).should('have.attr', 'aria-label');
    });

    // Check live regions
    cy.get('[aria-live="polite"]').should('exist');
    cy.get('[aria-live="assertive"]').should('exist');

    // Check form associations
    cy.get('input').each(($input) => {
      const id = $input.attr('id');
      cy.get(`label[for="${id}"]`).should('exist');
    });
  });
});
```

Manual Accessibility Checklist

```markdown
## Keyboard Navigation
- [ ] All interactive elements reachable via keyboard
- [ ] Tab order follows logical flow
- [ ] Focus visible at all times
- [ ] Escape key closes modals/popups
- [ ] Enter/Space activate buttons
- [ ] Arrow keys navigate menus

## Screen Reader Testing
- [ ] All content readable
- [ ] Form fields properly labeled
- [ ] Error messages announced
- [ ] Status changes announced
- [ ] Navigation landmarks present
- [ ] Headings properly structured

## Visual Testing
- [ ] Works at 200% zoom
- [ ] Responsive to font size changes
- [ ] High contrast mode compatible
- [ ] No information conveyed by color alone
- [ ] Focus indicators meet contrast requirements
```

7.3 Cross-Platform Testing

Device Matrix

Platform	Browsers	Versions	Test Coverage
iOS	Safari, Chrome	14+	Full E2E
Android	Chrome, Firefox	10+	Full E2E
Windows	Chrome, Edge, Firefox	Latest 2	Full E2E
macOS	Safari, Chrome, Firefox	Latest 2	Full E2E
Linux	Chrome, Firefox	Latest	Smoke tests

Responsive Testing

```javascript
```

```
// Viewport testing
const viewports = [
  { name: 'iPhone SE', width: 375, height: 667 },
  { name: 'iPhone 14', width: 390, height: 844 },
  { name: 'iPad', width: 768, height: 1024 },
  { name: 'iPad Pro', width: 1024, height: 1366 },
  { name: 'Desktop', width: 1920, height: 1080 },
  { name: '4K', width: 3840, height: 2160 }
];

viewports.forEach(viewport => {
  describe(`${viewport.name} (${viewport.width}x${viewport.height})`, () => {
    beforeEach(() => {
      cy.viewport(viewport.width, viewport.height);
    });

    it('should display correctly', () => {
      cy.visit('/');
      cy.screenshot(`${viewport.name}-homepage`);

      // Check layout
      if (viewport.width < 768) {
        cy.get('[data-testid="mobile-menu"]').should('be.visible');
        cy.get('[data-testid="sidebar"]').should('not.be.visible');
      } else {
        cy.get('[data-testid="mobile-menu"]').should('not.be.visible');
        cy.get('[data-testid="sidebar"]').should('be.visible');
      }
    });
  });
});
```

PWA Testing

```
javascript
```

```
// Service Worker tests
describe('PWA Functionality', () => {
  it('should install service worker', () => {
    cy.visit('/');
    cy.window().then((win) => {
      expect(win.navigator.serviceWorker.controller).to.not.be.null;
    });
  });

  it('should work offline', () => {
    cy.visit('/');
    cy.wait(2000); // Let SW cache assets

    // Go offline
    cy.window().then((win) => {
      win.navigator.onLine = false;
    });

    // Should still work
    cy.reload();
    cy.get('[data-testid="task-list"]').should('be.visible');
    cy.createTask('Offline task');

    // Go online
    cy.window().then((win) => {
      win.navigator.onLine = true;
    });

    // Should sync
    cy.get('[data-testid="sync-indicator"]').should('contain', 'Synced');
  });

  it('should be installable', () => {
    cy.visit('/');
    cy.window().then((win) => {
      let installPromptEvent;

      win.addEventListener('beforeinstallprompt', (e) => {
        installPromptEvent = e;
      });

      expect(installPromptEvent).to.not.be.undefined;
    });
```

```
  });
});
```

7.4 Security Testing

Security Test Suite

javascript

```
// Security vulnerability tests
describe('Security Tests', () => {
  it('should prevent XSS attacks', () => {
    const xssPayloads = [
      '<script>alert("XSS")</script>',
      '<img src=x onerror=alert("XSS")>',
      'javascript:alert("XSS")',
      '<svg onload=alert("XSS")>'
    ];

    xssPayloads.forEach(payload => {
      cy.createTask(payload);
      cy.get('[data-testid="task-list"]').should('not.contain', '<script>');
      cy.get('[data-testid="task-list"]').should('contain', payload.replace(/</g, '&lt;'));
    });
  });

  it('should prevent SQL injection', () => {
    const sqlPayloads = [
      "'; DROP TABLE tasks; --",
      "1' OR '1'='1",
      "admin'--"
    ];

    sqlPayloads.forEach(payload => {
      cy.request({
        method: 'POST',
        url: '/api/tasks',
        body: { title: payload },
        failOnStatusCode: false
      }).then((response) => {
        expect(response.status).to.be.oneOf([200, 201]);
        // Verify database still intact
        cy.request('/api/tasks').then((res) => {
          expect(res.status).to.equal(200);
        });
      });
    });
  });

  it('should enforce authentication', () => {
    cy.request({
      method: 'GET',
```

```
      url: '/api/tasks',
      headers: { Authorization: '' },
      failOnStatusCode: false
    }).then((response) => {
      expect(response.status).to.equal(401);
    });
  });

  it('should validate input', () => {
    const invalidInputs = [
      { title: '' }, // Empty title
      { title: 'a'.repeat(501) }, // Too long
      { priority: 11 }, // Invalid priority
      { due_date: 'not-a-date' } // Invalid date
    ];

    invalidInputs.forEach(input => {
      cy.request({
        method: 'POST',
        url: '/api/tasks',
        body: input,
        failOnStatusCode: false
      }).then((response) => {
        expect(response.status).to.equal(400);
        expect(response.body).to.have.property('error');
      });
    });
  });
});
```

Security Checklist

markdown

```
## Authentication & Authorization
- [ ] JWT tokens properly validated
- [ ] Refresh tokens rotate on use
- [ ] Session timeout implemented
- [ ] RBAC properly enforced
- [ ] Password requirements enforced

## Data Protection
- [ ] All data encrypted in transit (TLS)
- [ ] Sensitive data encrypted at rest
- [ ] PII properly masked in logs
- [ ] Secure cookie flags set
- [ ] CORS properly configured

## Input Validation
- [ ] All inputs sanitized
- [ ] File uploads validated
- [ ] Rate limiting implemented
- [ ] Request size limits enforced
- [ ] SQL injection prevented

## Security Headers
- [ ] CSP header configured
- [ ] X-Frame-Options set
- [ ] X-Content-Type-Options set
- [ ] Strict-Transport-Security enabled
- [ ] Referrer-Policy configured
```

7.5 Performance Testing

Load Testing Scenarios

```
javascript
```

```
// K6 load testing script
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  stages: [
    { duration: '2m', target: 100 }, // Ramp up
    { duration: '5m', target: 100 }, // Stay at 100
    { duration: '2m', target: 200 }, // Ramp to 200
    { duration: '5m', target: 200 }, // Stay at 200
    { duration: '2m', target: 0 },   // Ramp down
  ],
  thresholds: {
    http_req_duration: ['p(95)<500'], // 95% under 500ms
    http_req_failed: ['rate<0.01'],   // <1% error rate
    iteration_duration: ['p(90)<1000'] // 90% under 1s
  }
};

export default function() {
  // Login
  const loginRes = http.post('https://app.vercel.app/api/auth/login', {
    email: 'test@example.com',
    password: 'password'
  });

  check(loginRes, {
    'login successful': (r) => r.status === 200
  });

  const token = loginRes.json('access_token');
  const headers = { Authorization: `Bearer ${token}` };

  // Get tasks
  const tasksRes = http.get('https://app.vercel.app/api/tasks', { headers });
  check(tasksRes, {
    'tasks retrieved': (r) => r.status === 200
  });

  // Create task
  const createRes = http.post('https://app.vercel.app/api/tasks',
    JSON.stringify({ title: `Task ${Date.now()}` }),
    { headers }
```

```
  );
  check(createRes, {
    'task created': (r) => r.status === 201
  });

  sleep(1);
}
```

Stress Testing

```javascript
// Stress test configuration
export const stressOptions = {
  stages: [
    { duration: '5m', target: 500 },  // Ramp to 500 users
    { duration: '10m', target: 500 }, // Stay at 500
    { duration: '5m', target: 1000 }, // Push to 1000
    { duration: '10m', target: 1000 }, // Stay at 1000
    { duration: '5m', target: 0 },    // Ramp down
  ],
  thresholds: {
    http_req_duration: ['p(99)<2000'], // 99% under 2s
    http_req_failed: ['rate<0.05'],    // <5% error rate
  }
};
```

Document Version: 1.0.0
Last Updated: January 2024
Next Review: February 2024