سند ۴,۵: استراتژی و تست + شواهد اجرا QA — ShahnamehMap

نسخه: ۱,۰

تاریخ: ۱۵/۰۸/۱۴۰۳

تهیه‌کننده: مدیر فنی (CTO) / مسئول تضمین کیفیت (QA Lead)

وضعیت: فعال

---

۱. فلسفه QA: پیشگیری، نه بازرسی

ما QA را به عنوان **فرآیندی مستمر و تعبیه‌شده در چرخه توسعه** می‌بینیم، نه یک مرحله جداگانه در پایان. هدف، **کشف و جلوگیری از باگ در اولین فرصت ممکن** است، زیرا هزینه رفع آن به طور تصاعدی با گذشت زمان افزایش می‌یابد.

اصل کلی: "اگر تست نشده، شکسته است".

---

۲. استراتژی جامع تست (Testing Strategy Matrix)

ما از یک **هرم تست (Test Pyramid)** اصلاح‌شده با تمرکز بر اتوماسیون و تست‌های معنادار پیروی می‌کنیم.

```
                    ^ تعداد تست‌ها
                    |
        [ تست اکتشافی/دستی ]          |
                    /\              |
                   /  \             |
                  /    \            |
هدف: ** <- [ End-to-End (E2E) تست‌های ]          |
**جریان‌های حیاتی
                /      \            |
               /        \           |
```

هدف: ** -> [(Integration) تست‌های یکپارچگی] |
**تعامل سرویس‌ها

| / |
\ / \
| / |
\ / \
تست‌های کامپوننت] <-] [(Unit) تست‌های واحد] |
**پایه اصلی

| / |
\ / \

_______/_____|
سرعت اجرا <_____

```

**پایه هرم - (Unit Tests) تست واحد ۲,۱.** ###

هدف:** اعتبارسنجی عملکرد **کوچک‌ترین واحدهای کد** **  *
.**(توابع، کلاس‌ها، هوک‌ها) در **انزوا

، توابع محاسباتی، (Business Logic) حوزه:** منطق کسب‌وکار**  *
transformers.
```

.(Jest به دلیل سرعت بالاتر نسبت به) **Vitest** **:ابزار** *

هدف:** **≤ ۸۵٪ برای منطق کسب‌وکار** Coverage معیار** *
.(توابعی که تصمیمات محصول را پیاده‌سازی می‌کنند)

.(بخش ۴) CI در Vitest شواهد:** گزارش اجرای** *

Damage) مثال: تست واحد برای تابع محاسبه آسیب**
Calculation):**

```typescript
// src/logic/combat/damageCalculator.ts

export function calculateDamage(attackerStr: number, defenderArmor: number, weaponMultiplier: number): number {
  if (attackerStr <= 0) return 0;
  const baseDamage = attackerStr * weaponMultiplier;
  const mitigatedDamage = Math.max(baseDamage - defenderArmor, 1); // حداقل ۱ آسیب
  return mitigatedDamage;
}
```

```
// __tests__/logic/combat/damageCalculator.test.ts
import { describe, it, expect } from 'vitest';
import { calculateDamage } from
'@/logic/combat/damageCalculator';

describe('calculateDamage', () => {
it('should return 0 if attacker strength is 0 or negative',
() => {
expect(calculateDamage(0, 5, 1.5)).toBe(0);
expect(calculateDamage(-10, 5, 1.5)).toBe(0);
});
it('should calculate base damage correctly', () => {
expect(calculateDamage(10, 0, 2)).toBe(20); // 10 * 2
});
it('should apply armor mitigation correctly', () => {
expect(calculateDamage(10, 15, 2)).toBe(5); // (20 -
15) = 5
});
it('should ensure minimum damage of 1', () => {
```

حداقل // ;expect(calculateDamage(1, 100, 1)).toBe(1)
آسیب

});

});

```

### **۲,۲. تست کامپوننت (Component Tests)**

* **ابزار:** **Vitest** + **React Testing Library** (با فلسفه "تست از دید کاربر").

* **معیار Coverage:** هدف: ≥ ۷۰٪ برای کامپوننت‌های حیاتی (فرم‌ها، کامپوننت‌های تعاملی).

* **استراتژی Mock:** Mock کردن `fetch`/`axios` و های تست با ارائه React های context و state.

**مثال: تست کامپوننت `CharacterCreatorForm`:**

```typescript

```
// __tests__/components/CharacterCreatorForm.test.tsx
import { render, screen, fireEvent, waitFor } from '@testing-library/react';
import userEvent from '@testing-library/user-event';
import { CharacterCreatorForm } from '@/components/CharacterCreatorForm';
import { RaceProvider } from '@/contexts/RaceContext';

describe('CharacterCreatorForm', () => {
  it('should render all race and class options from context', () => {
    render(<RaceProvider><CharacterCreatorForm /></RaceProvider>);

    expect(screen.getByLabelText(/نژاد/i)).toBeInTheDocument();

    expect(screen.getByLabelText(/کلاس/i)).toBeInTheDocument();
  });
```

```
it('should disable submit button until all fields are valid', () => {
    render(<RaceProvider><CharacterCreatorForm /></RaceProvider>);
    expect(screen.getByRole('button', { name: /ساخت/i })).toBeDisabled();

    // شبیه‌سازی پر کردن فرم
    fireEvent.change(screen.getByLabelText(/نام/i), { target: { value: 'آرش' } });
    // ... پر کردن بقیه فیلدها
    waitFor(() => expect(screen.getByRole('button', { name: /ساخت/i })).toBeEnabled());
  });
});
```

۲,۳. تست یکپارچگی (Integration Tests)**

* **هدف:** اعتبارسنجی تعامل صحیح بین ماژول‌ها یا سرویس‌ها (مثلاً API بک‌اند و دیتابیس، دو سرویس).

، سرویس‌های دیتابیس، ارتباط با API Endpoints **:حوزه** *
.(**شده Mock** مثلاً درگاه پرداخت) سرویس‌های خارجی

+ (API برای تست) **Supertest** + **Vitest** **:ابزار** *
.(Dockerized PostgreSQL) **دیتابیس تست ایزوله**

هدف:** **٪۱۰۰ پوشش برای مسیرهای حیاتی Coverage معیار** *
.(کاربر، بازی، پرداخت) **API

:ایجاد کاراکتر API مثال: تست یکپارچگی

```typescript
// __tests__/api/characters.integration.test.ts
import request from 'supertest';
import { app } from '@/api/server';
import { prisma } from '@/lib/prisma';
import { clearTestDatabase, seedTestUser } from '@/tests/helpers';

describe('POST /api/v1/characters', () => {
  beforeEach(async () => {
```

```
  await clearTestDatabase();
  await seedTestUser(); // کاربر تست با توکن معتبر
});

it('should create a new character for authenticated user', async () => {
  const response = await request(app)
    .post('/api/v1/characters')
    .set('Authorization', 'Bearer test-user-valid-token')
    .send({
      name: 'رستم تست',
      race: 'پهلوان',
      class: 'جنگاور',
    });
  expect(response.status).toBe(201);
  expect(response.body).toHaveProperty('id');
  // تأیید ذخیره در دیتابیس
  const dbChar = await prisma.character.findFirst();
  expect(dbChar?.name).toBe('رستم تست');
```

```
});

it('should return 401 for unauthenticated requests',
async () => {

const response = await request(app)

.post('/api/v1/characters')

.send({ name: 'نامشروع' });

expect(response.status).toBe(401);

});

});
```

۲,۴. تست End-to-End (E2E)

* **هدف:** شبیه‌سازی **جریان کامل کاربر** در یک مرورگر واقعی، از رابط کاربری تا بک‌اند و دیتابیس.

* **حوزه:** **جریان‌های حیاتی کسب‌وکار:** ثبت‌نام → ساخت کاراکتر → شروع بازی → خرید اشتراک.

* **ابزار:** **Playwright** (به دلیل قابلیت‌های cross-browser قوی، سرعت و گزارش‌گیری).

* **معیار Coverage** ۱۰۰٪ پوشش برای جریان‌های حیاتی تعریف‌شده** (حداقل ۵ سناریو اصلی).

مثال: تست E2E کاراکتر اولین و ساخت "ثبت‌نام جریان":

```typescript
// tests/e2e/user-onboarding.spec.ts
import { test, expect } from '@playwright/test';

test('new user can sign up and create a first character', async ({ page }) => {
  // ۱. رفتن به صفحه اصلی
  await page.goto('https://staging.shahnamehmap.ir');

  // ۲. کلیک روی ثبت‌نام و پر کردن فرم
  await page.getByRole('button', { name: /ثبت‌نام/i }).click();

  await page.getByLabel(/ایمیل/i).fill('test@example.com');
  await page.getByLabel(/رمز عبور/i).fill('StrongPass123!');
```

```
await page.getByRole('button', { name: /ایجاد حساب/i }).click();

// ۳. انتظار برای هدایت به داشبورد
await expect(page).toHaveURL(/.*dashboard/);

// ۴. کلیک روی "ساخت پهلوان"
await page.getByRole('button', { name: /ساخت پهلوان/i }).click();

// ۵. پر کردن فرم کاراکتر
await page.getByLabel(/نام پهلوان/i).fill('آرش دیجیتال');
await page.locator('select[name="race"]').selectOption('ایرانی');

// ۶. تأیید ساخت و مشاهده کارت کاراکتر
await page.getByRole('button', { name: /بساز/i }).click();
await expect(page.locator('text=آرش دیجیتال')).toBeVisible();
});
```

۲,۵. تست امنیت (Security Testing)**

* **اتوماتیک:** **SAST (Static Application Security Testing)** با **GitHub Advanced Security / Snyk Code**.
برای شناسایی آسیب‌پذیری‌های رایج در کد CI در خط لوله.

* **نیمه‌اتوماتیک/دستی:**

* کنترل دسترسی** برای **(Authorization)** بررسی دستی **های حساس endpoint.

* تست **XSS** و **SQL** تزریق** با استفاده از Payload های استاندارد بر روی محیط Staging.

* **تست نفوذ** انجام **تست** سالی یک بار** یا پس از تغییرات عمده معماری** برون‌سپاری:** (Penetration Test)**.

۲,۶. تست رگرسیون (Regression Testing)**

* تعریف:** اطمینان از اینکه تغییرات جدید، عملکردهای موجود را **خراب نکرده‌اند**.

* **روش اجرا:**

اتوماتیک:** **تمام مجموعه تست‌های واحد، یکپارچگی و** ۱.
.اجرا می‌شوند. این سد اصلی رگرسیون است commit در هر **E2E

، یک (Release) دستی ساختاریافته:** قبل از هر انتشار اصلی** ۲.
توسط تیم Staging **چک‌لیست تست رگرسیون دستی** روی محیط
.اجرا می‌شود. این چک‌لیست شامل ۳۰-۲۰ نقطه کلیدی از سیستم است QA

QA و نقش (Release Process) فرآیند انتشار .۳ ##

.انتشار ما **چندمرحله‌ای** و **گیت‌محور** است

```mermaid
graph LR
A[در توسعه feature/*] -->|PR + تست‌های CI| B(مرج به develop);
B --> C[استقرار خودکار در Staging];
C --> D[تست رگرسیون دستی + تست پذیرش];
```

D -->|تأیید QA Lead| E[مرج به main];

E --> F[ایجاد Git Tag];

F --> G[استقرار خودکار در Production];

G --> H[نظارت بر لاگ و متریک];

```

* **دروازه‌های کیفیت (Quality Gates):**

۱. **Gate 1 (PR):** عبور تمام تست‌های واحد، یکپارچگی و لینت.

۲. **Gate 2 (Pre-Staging):** عبور تست‌های E2E روی محیط تست.

۳. **Gate 3 (Pre-Production):** تأیید دستی QA Lead بر اساس نتایج تست رگرسیون در Staging.

* **Rollback Plan:** در صورت مشاهده مشکل بحرانی در Production در ۳۰ دقیقه اول، امکان بازگشت خودکار به نسخه قبلی از طریق CI/CD وجود دارد.

---
```

۴. شواهد اجرا و گزارشات (Test Evidence)

۴,۱. گزارش اجرای تست در CI (GitLab CI):

```yaml
# نمونه خروجی مرحله test در CI Pipeline

Unit & Component Tests:

✓ Vitest passed.

- Tests: 1245 passed, 0 failed

- Coverage: 89% (Lines) | 85% (Functions) | 82% (Branches)

- Time: 45s

Integration Tests:

✓ API Integration passed.

- Tests: 87 passed, 0 failed

- Time: 12s
```

E2E Tests (Playwright):

✓ User Onboarding passed.

✓ Character Creation passed.

✓ Game Session Start passed.

- Tests: 5 passed, 0 failed

- Time: 1m 30s

```

*(قابل مشاهده GitLab CI این گزارش به صورت گرافیکی در داشبورد است.)*

### **۴,۲. داشبورد پوشش کد (Code Coverage Dashboard):**

ما از **Vitest UI** و **گزارش‌های HTML** برای مشاهده پوشش کد استفاده می‌کنیم. هر توسعه‌دهنده می‌تواند قبل از ارسال PR، پوشش کد تغییرات خود را ببیند.
```

(combat, turn-management) نمودار نشان می‌دهد پوشش توابع منطق بازی* *.بالای ۹۰٪ است

۴,۳. گزارش تست رگرسیون دستی:

به‌روز که نتایج اجرای **Google Sheet/ Notion Table** یک چک‌لیست رگرسیون قبل از هر انتشار را ثبت می‌کند. ستون‌ها: `Feature`, `Test Scenario`, `Tester`, `Status (Pass/Fail/Blocked)`, `Bug ID (if any)`, `Notes`.

---

۵. سیاست رفع باگ (Bug Triage & Resolution)

۵,۱. طبقه‌بندی و اولویت‌بندی باگ‌ها:

اولویت	**تعریف**	**زمان پاسخ هدف**	**نمونه**

| **P0 (Critical)** | سیستم کاملاً از کار افتاده. از دست رفتن داده. | < ۲ ساعت | کاربران نمی‌توانند وارد سیستم شوند. پرداخت موفق، اما اشتراک فعال نمی‌شود. |

| **P1 (High)** | عملکرد اصلی مختل شده، اما راه‌کار موقت وجود دارد. | < ۸ ساعت (روز کاری) | ساخت کاراکتر جدید با یک کلاس خاص با خطا مواجه می‌شود. |

| **P2 (Medium)** | مشکل در عملکرد غیراصلی یا UI. راه‌کار وجود دارد. | < ۴۸ ساعت | ترازو نمایش اشتباه در صفحه کاراکتر |

| **P3 (Low)** | درخواست بهبود جزئی، مشکل typo، زیبایی‌شناختی. | در اسپرینت بعدی | فونت در یک بخش کوچک ناهماهنگ است. |

۵,۲. **فرآیند رفع باگ:**

۱. **گزارش:** از طریق **گزارش خطا در برنامه** (یا Sentry) **تیکت پشتیبانی**.

۲. **Triaging:** جلسه روزانه تیم توسعه/QA برای **تخصیص اولویت و مالک** به باگ‌های جدید.

۳. **تست برای باگ را رفع و باگ را توسعه‌دهنده مالک، رفع:** بازتولید (Reproduce) می‌کند. آن را می‌نویسد تا از بازگشت (Regression) جلوگیری کند.

۴. **تأیید:** QA تأیید می‌کند که باگ رفع شده و رگرسیون جدیدی ایجاد نکرده است.

۵. **بستن:** باگ در سیستم ردیابی (Jira) بسته می‌شود.

۵,۳. معیارهای کیفی (Quality Metrics) تحت نظارت:**

* **Defect Density:** (تعداد باگ‌های کشف‌شده در یک دوره / تعداد story point یا تعداد خط کد). هدف: کاهش روندی.

* **Bug Leakage Ratio:** (تعداد باگ‌های کشف‌شده در Production) / (کل باگ‌های کشف‌شده). هدف: < ۵٪.

* **Mean Time To Resolution (MTTR):** میانگین زمان رفع باگ‌های P0/P1. هدف: < ۱۲ ساعت.

---

۶. نتیجه: تضمین کیفیت به عنوان یک ضمانت رشد**

به ما اطمینان می‌دهد که هر ویژگی جدید QA این استراتژی:

1. **قبل از ادغام**، از نظر منطق پایه معتبر است (تست واحد).

2. **قبل از استقرار**، با بقیه سیستم سازگار است (تست یکپارچگی و E2E).

3. **قبل از انتشار**، از طریق دید کاربر نهایی بررسی شده است (تست رگرسیون دستی).

4. **پس از انتشار**، تحت نظارت است و باگ‌هایش به سرعت ردیابی و رفع می‌شوند.

نبود این چارچوب به معنای پذیرش ریسک خرابی سیستم با هر تغییر، **افزایش تصاعدی هزینه رفع اشکال، و در نهایت کاهش شدید سرعت تحویل (Velocity)** ، سرمایه‌گذاری روی QA تیم است.** سرمایه‌گذاری روی **قابلیت پیش‌بینی و سرعت رشد پایدار** است.