Credit Gyems Academy - Comprehensive Testing Guide

📋 Table of Contents

🎯 Overview

The Credit Gyems Academy testing suite provides comprehensive coverage across all aspects of the application:

- **GUI Testing**: Visual interface validation, user interactions, responsive design
- **API Testing**: Endpoint functionality, data validation, authentication
- **Edge Case Testing**: Payment scenarios, concurrency, security, data integrity
- **Performance Testing**: Load testing, stress testing, response times
- **Accessibility Testing**: WCAG compliance, keyboard navigation, screen readers

Testing Philosophy

- **100% Critical Path Coverage**: All user journeys must work flawlessly
- **Edge Case Resilience**: Handle unexpected scenarios gracefully
- **Performance Standards**: Meet or exceed industry benchmarks
- **Accessibility First**: Ensure usability for all users

🏗️ Testing Architecture

```
credit-gyems-academy/
├── scripts/TS_CGA_v1/           # PowerShell test scripts
│   ├── Test-Utilities.ps1       # Common testing utilities
│   ├── Run-CompleteSuite.ps1    # Master test runner
│   ├── Run-GUITests.ps1         # GUI test runner
│   ├── Test-*.ps1               # Individual test suites
│   └── k6/                      # K6 stress test scripts
├── gui-tests/                   # Playwright GUI tests
│   ├── navigation.spec.js       # Navigation & routing tests
│   ├── form-validation.spec.js  # Form validation tests
│   ├── responsive-design.spec.js # Responsive design tests
│   ├── user-flows.spec.js       # E2E user journey tests
│   ├── accessibility.spec.js    # Accessibility tests
│   ├── visual-regression.spec.js # Visual consistency tests
│   ├── performance.spec.js      # Performance tests
│   └── component-testing.spec.js # React component tests
├── test-reports/                # Generated test reports
├── playwright.config.js         # Playwright configuration
└── .github/workflows/           # CI/CD workflows
    ├── gui-tests.yml           # GUI testing workflow
    └── edge-case-tests.yml     # Edge case testing workflow
```

🖥️ GUI Testing Suite

Overview

The GUI testing suite uses **Playwright** for cross-browser testing and covers:

- Navigation flows
- Form interactions and validation
- Responsive design across devices
- Visual regression testing
- Accessibility compliance
- Performance metrics

Test Categories

1. Navigation Tests (`navigation.spec.js`)

- Main menu navigation
- 404 page handling
- Navigation state persistence
- Loading states during transitions

2. Form Validation (`form-validation.spec.js`)

- Registration form validation
- Credit card input formatting
- Real-time validation feedback
- Error message display
- Date/time picker validation

3. Responsive Design (`responsive-design.spec.js`)

- Mobile (375px), Tablet (768px), Desktop (1920px) viewports
- Orientation changes
- Touch interactions
- Grid layout adjustments

4. User Flows E2E (`user-flows.spec.js`)

Complete user journeys:

- **Purchase Flow**: Browse → Add to Cart → Checkout → Confirmation
- **Booking Flow**: Select Service → Choose Date/Time → Confirm
- **Community Flow**: Create Discussion → Post Reply → Like/Unlike

5. Accessibility (`accessibility.spec.js`)

- WCAG 2.1 compliance
- Keyboard navigation
- Screen reader compatibility
- Color contrast validation
- Focus management

6. Visual Regression (`visual-regression.spec.js`)

- Screenshot comparisons
- Component visual consistency
- Dark mode support
- Hover/focus states

7. Performance (`performance.spec.js`)

- Page load times
- Interaction responsiveness
- Memory leak detection
- Resource optimization

Running GUI Tests

```powershell
# Run all GUI tests
.\scripts\TS_CGA_v1\Run-GUITests.ps1

# Run specific browser
.\scripts\TS_CGA_v1\Run-GUITests.ps1 -Browser firefox

# Run in headless mode
.\scripts\TS_CGA_v1\Run-GUITests.ps1 -Headless

# Update visual snapshots
.\scripts\TS_CGA_v1\Run-GUITests.ps1 -UpdateSnapshots

# Run specific test file
.\scripts\TS_CGA_v1\Run-GUITests.ps1 -TestPattern "navigation"
```

Browser Support Matrix

Browser	Windows	macOS	Linux
Chrome	✅	✅	✅
Firefox	✅	✅	✅
Safari	❌	✅	❌
Edge	✅	✅	✅

🔌 API Testing Suite

The API testing suite validates all backend endpoints and includes:

- Authentication flows
- CRUD operations
- Data validation
- Error handling
- Authorization checks

Running API Tests

```powershell
# Run complete API test suite
.\scripts\TS_CGA_v1\Run-CreditGyemsQA.ps1

# Skip server startup (if already running)
.\scripts\TS_CGA_v1\Run-CreditGyemsQA.ps1 -SkipServerStart

# Clean up all test data after
.\scripts\TS_CGA_v1\Run-CreditGyemsQA.ps1 -CleanupAll
```

⚡ Edge Case Testing

Payment Edge Cases

- Partial refunds
- BNPL (Klarna/AfterPay) integration
- Payment failures and retries
- Subscription changes
- Currency conversions
- Chargebacks

Concurrent User Scenarios

- Multiple device logins
- Race conditions (inventory/bookings)
- Session conflicts
- Role changes while active

Security Testing

- Session hijacking prevention
- CSRF protection
- XSS/injection prevention
- Authorization bypasses

Data Integrity

- Orphaned records
- Cascade deletes
- Transaction atomicity
- Data retention compliance

Running Edge Case Tests

powershell

```powershell
# Quick edge case tests (5-10 min)
.\scripts\TS_CGA_v1\Run-AllEdgeCaseTests.ps1 -QuickRun

# Standard edge cases (30-45 min)
.\scripts\TS_CGA_v1\Run-AllEdgeCaseTests.ps1

# Full suite with destructive tests (60+ min)
.\scripts\TS_CGA_v1\Run-AllEdgeCaseTests.ps1 -FullSuite
```

🚀 Running Tests

Master Test Runner

The master test runner (`Run-CompleteSuite.ps1`) orchestrates all test suites:

```powershell
# Quick validation (API + basic GUI)
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode Quick

# Standard testing (API + GUI + edge cases)
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode Standard

# Full suite (everything including stress tests)
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode Full

# GUI tests only
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode GUI

# Edge cases only
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode Edge

# Everything with all browsers
.\scripts\TS_CGA_v1\Run-CompleteSuite.ps1 -TestMode All -Browser all
```

Test Modes Comparison

Mode	API Tests	GUI Tests	Edge Cases	Stress Tests	Duration
Quick	✅	Basic	❌	❌	5-10 min
Standard	✅	✅	Quick	❌	30 min
Full	✅	✅	✅	✅	60+ min
GUI	❌	✅	❌	❌	20 min
Edge	❌	❌	✅	Optional	45 min
All	✅	All browsers	✅	✅	2+ hours

🔄 CI/CD Integration

GitHub Actions Workflows

1. **GUI Tests** (`.github/workflows/gui-tests.yml`)
   - Triggers on frontend changes
   - Cross-browser testing matrix
   - Visual regression checks
   - Accessibility validation
2. **Edge Case Tests** (`.github/workflows/edge-case-tests.yml`)
   - Nightly runs
   - Full security scanning
   - Performance benchmarking

Setting Up CI/CD

1. **Configure Secrets**:

   ```
   TEST_MONGODB_URI
   TEST_JWT_SECRET
   TEST_STRIPE_SECRET
   TEST_SENDGRID_API_KEY
   SLACK_WEBHOOK (optional)
   ```

2. **Enable Workflows**:
   - Go to Actions tab in GitHub
   - Enable workflows
   - Configure branch protection rules
3. **Test Reports**:
   - HTML reports uploaded as artifacts
   - PR comments with test summaries
   - Slack notifications for failures

📊 Test Reports

Report Types

1. **HTML Reports**: Interactive dashboards with charts
2. **JSON Summaries**: Machine-readable results
3. **Screenshots/Videos**: For failed GUI tests
4. **Performance Metrics**: Response times, resource usage

Accessing Reports

powershell

```
# Reports are saved to:
test-reports/
├── complete-test-report-[timestamp].html
├── gui-test-results.json
├── edge-case-summary-[timestamp].json
├── payment-edge-cases-[timestamp].json
└── k6-stress-results/
```

💡 Best Practices

Writing New Tests

1. **GUI Tests**:

   javascript

   ```
   test('should validate user input', async ({ page }) => {
     // Arrange
     await page.goto('/form');

     // Act
     await page.fill('input[name="email"]', 'invalid');

     // Assert
     await expect(page.locator('.error')).toBeVisible();
   });
   ```

2. **API Tests**:

   powershell

   ```
   $result = Test-APIEndpoint `
     -Method "POST" `
     -Endpoint "http://localhost:5000/api/endpoint" `
     -Body @{ key = "value" } `
     -ExpectedStatus "200"
   ```

3. **Component Tests**:

   javascript

   ```
   test('Button renders correctly', async ({ mount }) => {
     const component = await mount(<Button>Click Me</Button>);
     await expect(component).toContainText('Click Me');
   });
   ```

Test Data Management

- Use unique timestamps for test data
- Clean up after tests
- Don't rely on specific database state
- Use factories for complex data

Performance Guidelines

- Page load < 3 seconds
- API response < 500ms (p95)
- First contentful paint < 1.5s
- No memory leaks

🔧 Troubleshooting

Common Issues

1. **Playwright Installation**:

   bash

   ```
   npx playwright install --with-deps
   ```

2. **Port Conflicts**:

   powershell

   ```
   # Check what's using ports
   netstat -ano | findstr :3000
   netstat -ano | findstr :5000
   ```

3. **Test Timeouts**:
   - Increase timeout in `playwright.config.js`
   - Check for slow API responses
   - Verify server is running

4. **Visual Regression Failures**:
   - Update snapshots if changes are intentional
   - Check for dynamic content that should be masked
   - Ensure consistent test environment

Debug Mode

```powershell
# Run with headed browser for debugging
$env:PWDEBUG=1
npx playwright test --headed

# Run specific test with debug
npx playwright test navigation.spec.js --debug
```

Getting Help

1. Check test logs in `test-reports/`
2. Review screenshots/videos for GUI failures
3. Check server logs for API issues
4. Consult team documentation

📈 Performance Benchmarks

Target metrics based on industry standards:

Metric	Target	Critical
Page Load Time	< 2s	> 3s
Time to Interactive	< 3.8s	> 5s
API Response (p95)	< 500ms	> 3000ms
Error Rate	< 0.5%	> 5%
Concurrent Users	1000+	< 500

🎯 Conclusion

This comprehensive testing suite ensures Credit Gyems Academy delivers a reliable, performant, and accessible experience for all users. Regular testing catches issues early and maintains the high quality standards expected by Coach Tae's audience.

Remember: **Quality is not an act, it's a habit** - Test early, test often, test thoroughly!