# GemeinceConnect - Test Strategy

Contents

[GemeinceConnect - Test Strategy 1](#_Toc201656018)

[1. Test Strategy 2](#_Toc201656019)

[1.1 Objective 2](#_Toc201656020)

[1.2 Scope 2](#_Toc201656021)

[1.3 Testing Levels 2](#_Toc201656022)

[1.4 Test Types 2](#_Toc201656023)

[1.5 Responsibilities 2](#_Toc201656024)

[1.6 Tools & Frameworks 2](#_Toc201656025)

[1.7 Module Coverage Matrix 2](#_Toc201656026)

[1.8 Entry/Exit Criteria 3](#_Toc201656027)

[2. Acceptance Test Strategy 3](#_Toc201656028)

[2.1 Objective 3](#_Toc201656029)

[2.2 Acceptance Criteria Template 3](#_Toc201656030)

[2.3 Traceability 3](#_Toc201656031)

[2.4 Format & Execution 3](#_Toc201656032)

# 1. Test Strategy

## 1.1 Objective

To define the structured approach to testing for the GemeindeConnect solution, including scope, types of testing, tools, responsibilities, and quality goals.

## 1.2 Scope

- Frontend: Form Designer UI, Template Editor, Form Preview, Submission View  
- Backend: API Endpoints, Workflow Logic, Data Storage  
- Integration: Authentication, Document Storage, E-Mail Service  
- Non-functional: Performance, Security (authentication, data protection)

## 1.3 Testing Levels

|  |  |  |
| --- | --- | --- |
| Level | Description | Tools/Frameworks |
| Unit | Test individual components and methods | xUnit |
| Integration | Test interaction between system components | Postman, SQL scripts |
| Acceptance | Business scenario validation | Manual, Gherkin scenarios |

## 1.4 Test Types

- Functional Testing  
- Regression Testing  
- Performance Testing  
- Security and Access Control Testing  
- Usability Testing (as part of UAT)

## 1.5 Responsibilities

|  |  |
| --- | --- |
| Role | Responsibility |
| Developer | Unit and integration tests |
| QA Engineer | Test case design, execution, automation |
| Product Owner | Acceptance test validation |
| Security Lead | Vulnerability scanning and access tests |

## 1.6 Tools & Frameworks

- Frontend: Manual  
- Backend: xUnit   
- Automation: Azure Pipelines

## 1.7 Module Coverage Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Test Level | Responsible | Notes |
| Form Designer UI | Unit, E2E | Frontend Team | UI components & rendering |
| Form API | Unit, Integration | Backend Team | Form save/load operations |
| Workflow Processor | Integration, System | QA + Dev | Condition evaluation tests |
| Auth Integration | Integration, Security | QA + Security | Azure B2C role checks |
| DB Layer | Unit, Regression | Dev + QA | EF Core + stored procs |

## 1.8 Entry/Exit Criteria

- Entry: Requirements approved, environment ready, test data prepared  
- Exit: All critical bugs resolved, tests passed, stakeholder sign-off

# 2. Acceptance Test Strategy

## 2.1 Objective

To validate that the delivered software meets business requirements through structured scenarios.

## 2.2 Acceptance Criteria Template

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Feature | Scenario | Preconditions | Steps | Expected Result |
| AT-001 | 1 | Create a new form with text field | User is logged in | Navigate to Designer > Add Field > Save | Form saved with text field |
|  |  |  |  |  |  |

## 2.3 Traceability

Each acceptance test links to a requirement or user story to ensure full coverage.

## 2.4 Format & Execution

Acceptance tests will be executed during UAT sessions with real users or proxy testers. Tests will be documented in a table or test management tool.