Fpt University

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REPORT

Report 5 – Software Test Documentation

– Hanoi, August 2025 –

Table of Contents

I. Record of Changes

Date	A* M, D	In charge	Change Description
15/11/2025	A	DucNV	Testing Documentation - part 1 (Scope of Testing) - Testing Targets is added
30/11/2025	M	CuongNT	Testing Documentation - part 2 (Test Strategy) is added
30/11/2025	A	CuongNT	Testing Documentation - part 3 (Test Plan) is added
30/11/2025	M	CuongNT	Testing Documentation - part 3 is modified
30/11/2025	M	CuongNT	Testing Documentation - part 2 is modified
30/11/2025	A	DucNV	Testing Documentation - part 4 (Test Cases) is added
1/12/2025	M	DucNV	Testing Documentation - part 4 is modified
1/12/2025	A	DucNV	Testing Documentation - part 5 (Test Reports) is added
1/12/2025	M	DucNV	Testing Documentation - part 5 (Test Reports) is modified
1/12/2025	M	DucNV	Testing Documentation - part 1 - Testing Levels is modified

*A - Added M - Modified D - Deleted

II. Testing Documentation

1. Scope of Testing

1.1. Testing Targets

1.1.1. Feature, Functional

The test scope of the project includes all features – functions defined in [Report 1_Project Introduction]

1.1.2. Non-Functional

Scope

- Backend (ASP.NET Core) and PostgreSQL.
- Includes APIs, background services, DB interactions, logging, configuration, deployment, and monitoring.

Availability

- Graceful Shutdown: Background services must respect CancellationToken for safe shutdown (the current service uses stoppingToken).
- Rollback: DB changes must have rollback plans and use transactions where appropriate.

Reliability & Consistency

- Atomic Updates: Payment status updates should be atomic; use transactions or compensating logic if multiple tables are modified.
- Retry Policy: DB connection retries are configured (EnableRetryOnFailure); background jobs should implement retry with backoff on transient failures. Current code logs exceptions and waits 5 minutes — consider structured retry/backoff and alerting.
- Acceptance Criteria: On temporary DB outage, job retries at least 3 times with exponential backoff and avoids duplicate updates.

Security

- Auth/Z: JWT authentication is used; ensure secret rotation and secure storage.
- Data in Transit/At Rest: Use TLS for external communications and consider encryption for sensitive fields.
- Acceptance Criteria: No sensitive secrets committed to repo; secret-scan CI passes.

Deployment & Configuration:

- Config-Driven: Important parameters (interval, enable/disable flags, batch size) configurable via appsettings or environment variables.
- Acceptance Criteria: Service can be disabled via configuration without code changes or redeployment.

2. Test Strategy

We determine that Agile testing is the best option for our project's testing procedure, given the specifics of that project. This test methodology offers the ideal stages for our project, from test preparation to test execution to test conclusion.

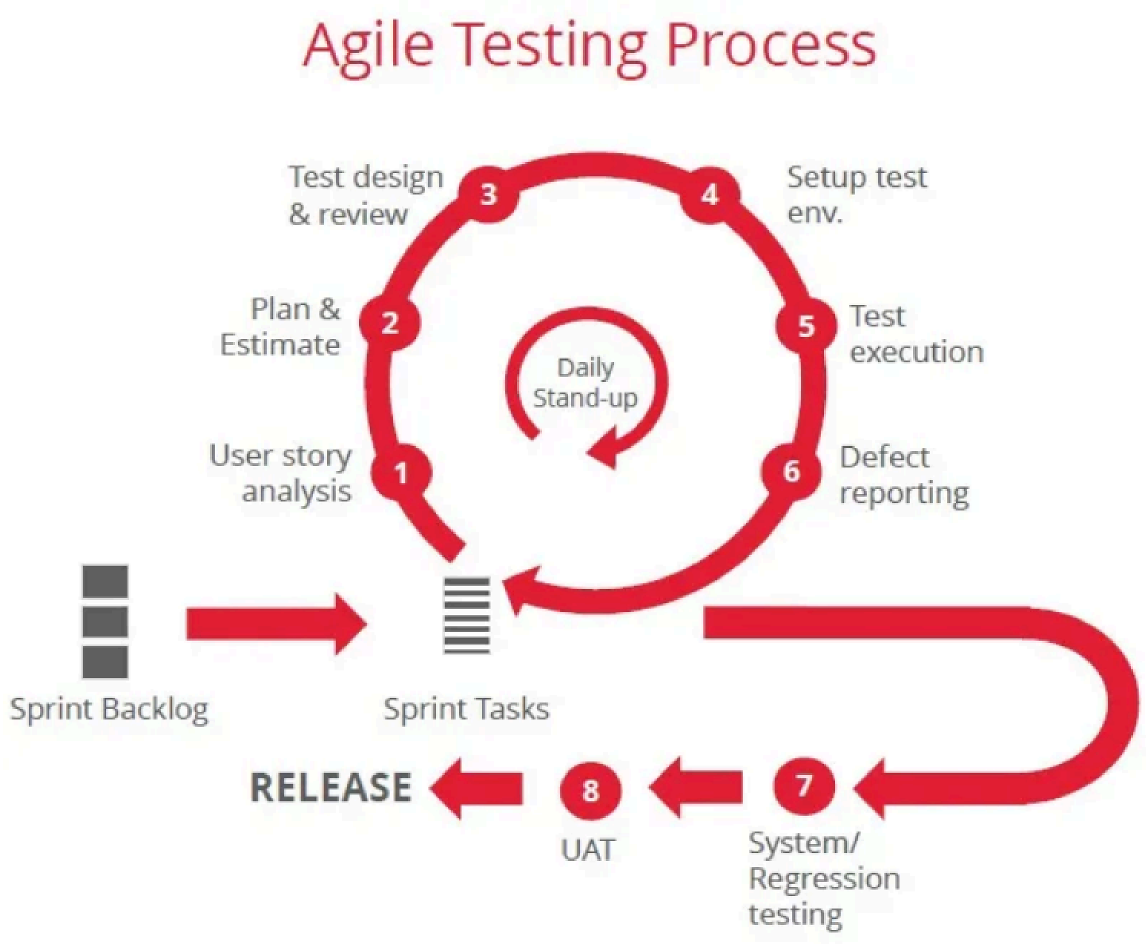


Figure 2.1. Agile Testing Process

No	Objective	Techniques	Completion criteria	Test level	Frequency
1	Verify small units of code (methods, classes) behave correctly in	Mock dependencies (repositories, DB context, external services) with	All unit tests pass in CI; minimum coverage target for core modules.	Unit	Developers

	isolation (business rules, utilities)	test doubles; assert outputs, exceptions, and state changes.			
2	Verify interactions between components, especially EF Core and PostgreSQL behavior, DI scopes, repository implementations, and transaction correctness.	Seed data, run code paths. Also test wiring of hosted service registration if needed.	Integration tests pass in CI stage; no unintended side effects on other tables.	Integration	Testers
3	Validate controller endpoints, request/response contracts, validation and auth enforcement.	Superficial startup in-memory server or hitting staging endpoints with test accounts; mock external calls.	All endpoints respond with expected status codes/data for defined inputs; authentication and authorization enforced per policy.	System	Testers
4	Validate full workflows with real integrations	Deploy to staging; use automated scripts to simulate user flows; include real or sandbox credentials for Cloudinary/Sepay or use service mocks where necessary.	All critical scenarios complete successfully; background job processed expected records; no regression on API beyond threshold.	Acceptance	End-users

2.1 Testing Types

Types	**Objective**	**Technique**	**Completion Criteria**
Unit Test	Verify the correctness of individual units of code in isolation so logic errors are caught early and cheaply.	Use a unit test framework. Replace external dependencies with mocks or fakes . Write focused tests for normal cases, edge cases, error paths, and parameterized inputs; include boundary and negative tests. Run fast and locally/CI on every commit.	All unit tests pass in CI; agreed coverage threshold for core modules met; new PRs do not introduce failing or flaky unit tests; execution time stays within fast-feedback limits.
GUI Testing	Validate user-facing UI behavior, layout, and usability so the application delivers the intended user experience across supported browsers/devices.	Combination of automated end-to-end UI tests for critical flows and manual exploratory/usability testing for edge-case UX. Use visual-regression tools for layout diffs, cross-browser runs, and test accounts or staging backends. Isolate flaky selectors and use stable locators.	All critical UI flows pass automated checks in CI/staging; visual-regression differences are within acceptable thresholds; cross-browser smoke matrix passes for supported browsers; no open high-severity UI defects.
API Testing	Ensure API endpoints meet contract, authentication, validation, error handling, and data correctness expectations	Automated contract tests Postman covering positive, negative, boundary, and security scenarios. Validate JSON schema, status codes, headers, and auth/permission enforcement. Run tests against ephemeral DB or staging.	All API contract tests in CI pass; authentication/authorization enforced for protected endpoints; backwards-compatibility checks for published contracts; endpoints return expected payloads and status codes for representative scenarios.
Non-functional Testing	Verify the system meets NFRs such as performance, scalability, reliability, availability, security, and maintainability	Run performance/load tests for throughput and latency; stress and soak tests for stability; security scans and secret scanning; reliability tests including chaos/injection if needed.	System survives stress/soak within defined degradation behavior; no unresolved critical/ high security findings; monitoring dashboards and alerts verified;

Regression Testing	Ensure that new code changes do not break existing functionality across the system — protect previously validated behavior.	Maintain an automated regression suite that includes a combination of unit, integration, and E2E tests. Use selective running, and nightly/full-suite execution in CI. Prioritize tests by risk and impact.	Regression suite passes before merging major changes or releasing; any test failures are triaged and fixed before release; acceptable flakiness rate with flaky tests either fixed or quarantined.
Database Testing	Verify database schema correctness, data integrity, migration safety, query performance, and that DB interactions behave as expected in realistic conditions. Ensure migrations apply cleanly and do not corrupt or lose data.	Use automated integration tests against an isolated PostgreSQL instance seeded with representative test data.	Migrations apply cleanly in staging; integrity checks pass; queries meet performance criteria; backup/restore completes within RTO and RPO targets.
UI/UX Testing	Validate that the product is usable, intuitive, and meets user expectations and accessibility standards.	Usability testing with representative users , heuristic evaluations, accessibility audits, analytics review.	Users complete target tasks with acceptable success/time/error rates; identified usability issues are addressed or logged with priorities; accessibility violations reduced to acceptable levels.

2.2 Test Levels

Type of Tests	**Test Level**			
	Unit	**Integration**	**System**	**Acceptance**
Unit Test	X			
GUI Test			X	
API Test		X	X	X
Non-functional Test			X	X
Regression Test	X	X	X	X
Database Test		X	X	
UI/UX Test			X	X

2.3 Supporting Tools

Purpose	Tool	Vendor/In-house	Version
Manage API test collections and run automated API tests	Postman	Postman Inc.	Postman latest
Manage test/staging database instances and versioned schema	PostgreSQL	PostgreSQL Global	PostgreSQL 18

3. Test Plan

3.1 Test Environment

Purpose	Tool	Provider
Unit test documents	Excel	Microsoft
Integration test documents	Excel	Microsoft
System test documents	Excel	Microsoft
Run Unit Test	Visual Studio	Microsoft
Run Integration Test	Postman	QA Team
Run System Test	Android Emulator / Real Device	QA Team
Testing tracking	GitHub, Excel	Project Team

3.2 Test Milestones

Milestone Task	Start Date	End Date
Create test plan	06/8/2025	09/8/2025
Create UAT test cases	22/8/2025	24/8/2025
Create ST test cases	25/8/2025	29/8/2025
Create and update UAT test cases	06/9/2025	08/9/2025
Create and update ST test cases	09/9/2025	10/9/2025
Create IT test cases	11/9/2025	12/9/2025
Create UT test cases	12/9/2025	13/9/2025
Create, update, and execute UT test cases	28/9/2025	03/10/2025

Create, update, and execute IT test cases	28/11/2025	03/11/2025
Create, update, and execute ST test cases	01/11/2025	03/11/2025
Create, update, and execute UAT test cases	01/11/2025	03/11/2025
Create, update, and execute UT test cases	14/11/2025	17/11/2025
Create, update, and execute IT test cases	14/11/2025	1711/2025
Create, update, and execute ST test cases	14/11/2025	17/11/2025
Create, update, and execute UAT test cases	14/11/2025	17/11/2025
Create, update, and execute UT test cases	30/11/2025	06/12/2025
Create, update, and execute IT test cases	30/11/2025	06/12/2025
Create, update, and execute ST test cases	30/11/2025	06/12/2025
Create, update, and execute UAT test cases	30/11/2025	06/12/2025
Full system testing	04/12/2025	06/12/2025
Create Test Report	04/12/2025	06/12/2025

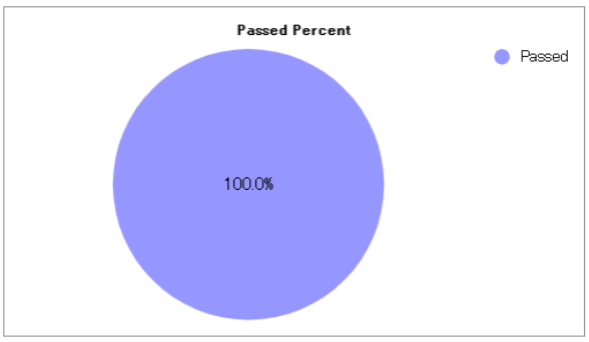
4. Test Cases

- Unit Test: SEP490_G151_UnitTest
- Integration test: SEP490_G151_IntegrationTest
- System test: SEP490_G151_SystemTest

5. Test Reports

Sub total	785	0	0	319	315	151	785

Test coverage	**100.00** %
Test successful coverage	**100.00** %
Normal case	**40.64** %
Abnormal case	**40.13** %
Boundary case	**19.24** %



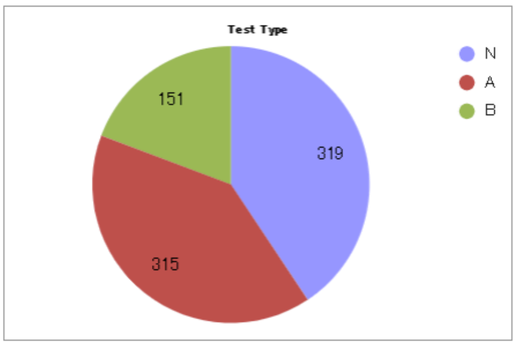
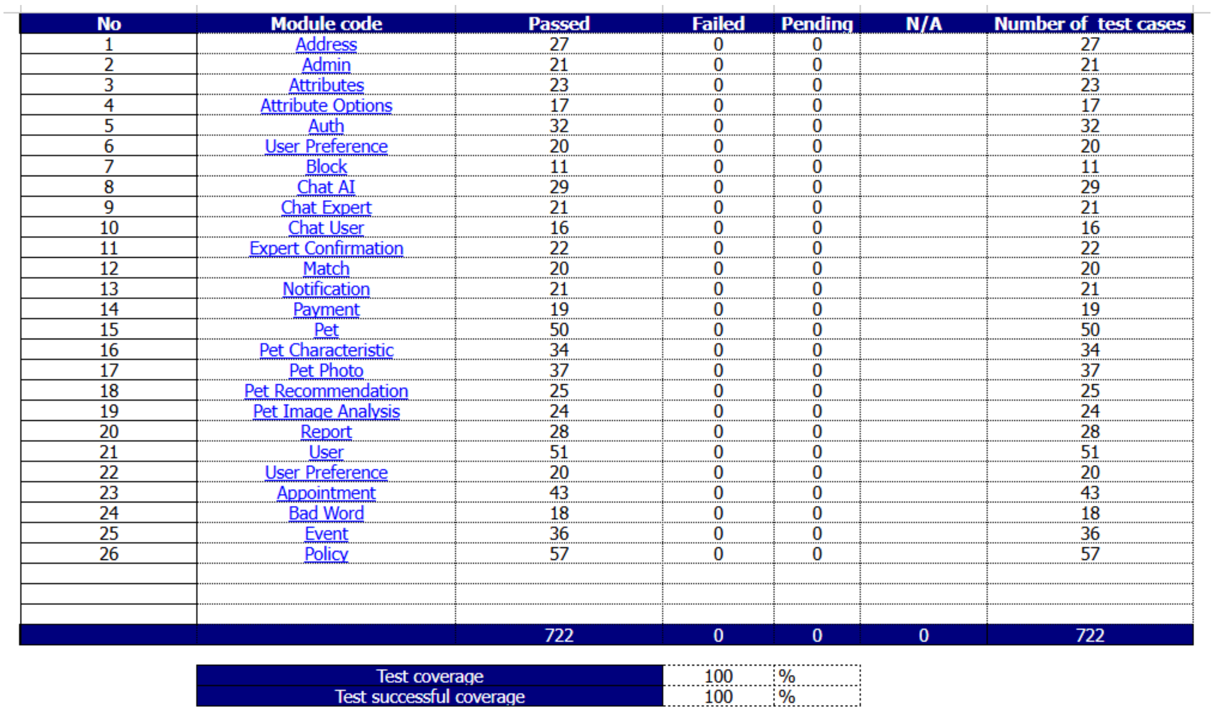


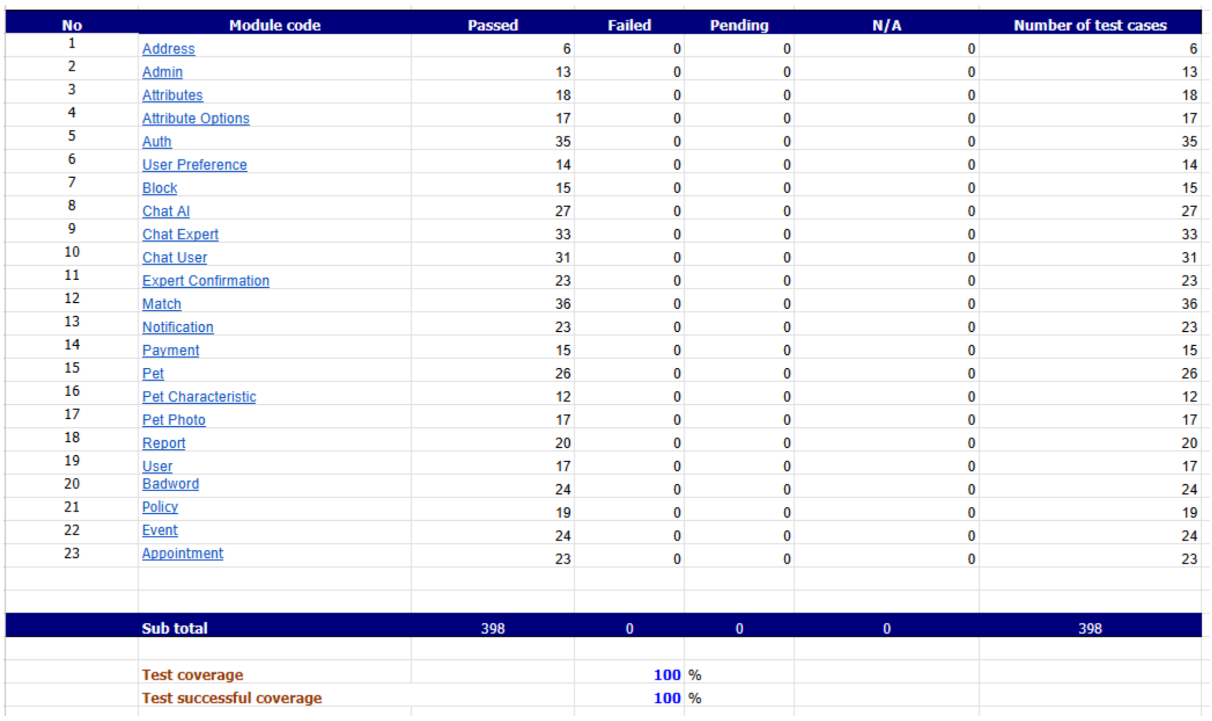
Figure 5.1: Unit Test Statistic



No	Module code	Passed	Failed	Pending	N/A	Number of test cases
1	Address	27	0	0		27
2	Admin	21	0	0		21
3	Attributes	23	0	0		23
4	Attribute Options	17	0	0		17
5	Auth	32	0	0		32
6	User Preference	20	0	0		20
7	Block	11	0	0		11
8	Chat AI	29	0	0		29
9	Chat Expert	21	0	0		21
10	Chat User	16	0	0		16
11	Expert Confirmation	22	0	0		22
12	Match	20	0	0		20
13	Notification	21	0	0		21
14	Payment	19	0	0		19
15	Pet	50	0	0		50
16	Pet Characteristic	34	0	0		34
17	Pet Photo	37	0	0		37
18	Pet Recommendation	25	0	0		25
19	Pet Image Analysis	24	0	0		24
20	Report	28	0	0		28
21	User	51	0	0		51
22	User Preference	20	0	0		20
23	Appointment	43	0	0		43
24	Bad Word	18	0	0		18
25	Event	36	0	0		36
26	Policy	57	0	0		57
		722	0	0	0	722

Test coverage	100	%
Test successful coverage	100	%

Figure 5.2: Integration Test Statistic



No	Module code	Passed	Failed	Pending	N/A	Number of test cases
1	Address	6	0	0	0	6
2	Admin	13	0	0	0	13
3	Attributes	18	0	0	0	18
4	Attribute Options	17	0	0	0	17
5	Auth	35	0	0	0	35
6	User Preference	14	0	0	0	14
7	Block	15	0	0	0	15
8	Chat AI	27	0	0	0	27
9	Chat Expert	33	0	0	0	33
10	Chat User	31	0	0	0	31
11	Expert Confirmation	23	0	0	0	23
12	Match	36	0	0	0	36
13	Notification	23	0	0	0	23
14	Payment	15	0	0	0	15
15	Pet	26	0	0	0	26
16	Pet Characteristic	12	0	0	0	12
17	Pet Photo	17	0	0	0	17
18	Report	20	0	0	0	20
19	User	17	0	0	0	17
20	Badword	24	0	0	0	24
21	Policy	19	0	0	0	19
22	Event	24	0	0	0	24
23	Appointment	23	0	0	0	23
	Sub total	398	0	0	0	398
	Test coverage		100	%		
	Test successful coverage		100	%		

Figure 5.3 System Test Statistic