Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee
Professor AlOmar
SSW 567
6 December 2024

TestPlanning

1. Introduction

1.1 Overall System and High-Level Goals

The Machine Readable Travel Document (MRTD) system is designed to facilitate the accurate and efficient processing of international travel documents. It comprises of functionalities for scanning, decoding, encoding, and validating travel document data, focusing on ensuring global compatibility and data integrity.

1.2 Testing Strategy Executive Summary

This project uses a Plan-Driven development approach. Testing is integrated throughout the project lifecycle to ensure alignment with specifications, achieve over 80% code coverage, and validate core functionalities like data encoding/decoding and check digit validation. Automated testing tools and structured reviews will mitigate risks and ensure quality.

1.3 Impact of Development Approach on Testing

The Plan-Driven approach emphasizes upfront planning, comprehensive requirement analysis, and structured implementation. This requires well-defined test cases aligned with functional specifications, and through reviews at each stage to minimize rework and ensure compliance with global standards.

2. Reference Other Relevant Documents

The ICAO document outlines the standard for creating Machine Readable Travel Documents (MRTDs), putting an emphasis on consistency and readability. A core requirement is the translation of non-Arabic numerical data in the Visual Inspection Zone into Arabic numerals to establish credibility across different systems. The document also places preferred requirements on the typeface, type size, font, and vertical line spacing, however the finalized requirements are determined by the issuing State or organization. The document also emphasizes using upper-case characters when defining names, however there is a permittance to using lower-case characters when a name is prefixed. Furthermore, the representation of dates and nationalities are standardized. Dates must be in accordance with the Gregorian calendar in this format DDnMMnYYYY. The representation of nationalities in MRTDs is standardized by the use of three letter country codes in the MRZ and the allowance for full textual representation in the

VIZ. If a non-latin language is used in the VIZ, then it must also be translated into an international language and separated with an oblique character (/).

3. Testing Scope

3.1 Scope of Testing

- **Testing Activities**
  - Decoding and encoding MRZ lines into structured fields
  - Validating check digits
  - Reporting mismatches in information fields
  - Performance testing for encoding/decoding large datasets
- **Exclusions**
  - Implementation of hardware components like scanners
  - Live database interactions (mocked for testing)

3.2 Prioritization Criteria

- Core functionalities (decoding, encoding, validation) have the highest priority
- Secondary focus on performance and edge case handling

4. Testing Approach

4.1 Key Factors

- Compliance with ICAO standards
- Accuracy of encoding/decoding processes
- Robustness in handling large datasets

4.2 Key Risks

- Misalignment with international standards
- Performance bottlenecks with large datasets
- Errors in check digit validation algorithms

4.3 Success Criteria

- ≥90% test coverage
- No critical defects in core functionalities
- Successful handling of edge cases and large datasets

4.4 Contingency Plans

- Incremental testing for performance issues
- Modular approach for independent testing of functionalities

4.5 Item Pass/Fail Criteria

- Tests pass if output matches expected results
- Failures identified and rectified in subsequent iterations

4.6 Entry/Exit Criteria

- Entry: All necessary modules are developed and code is ready for testing
- Exit: All test cases executed with ≥90% success rate

4.7 Testing Criteria and Checkpoints

- **Criteria**
    - All functionalities (decoding, encoding, validation) must perform as expected without critical defects
    - Code coverage must exceed 90%
    - All identified bugs must be resolved or deferred with a clear justification
    - Performance metrics for handling large datasets must meet acceptable thresholds
- **Checkpoints**
    - Completion of unit tests with 90%+ coverage
    - Verification of core functionalities in integration tests
    - Analysis of performance results for datasets of increasing size (100, 1000, 10000 records)
    - Approval of system tests covering end-to-end scenarios

4.8 Test Deliverables

- Test Case Documentation: Detailed descriptions of test cases, including input, expected output, and results
- Coverage Report: Generated by coverage.py, highlighting which parts of the code were executed during tests
- Mutation Testing Report: Results of MutPy, including number of mutants generated, killed, and survived
- Performance Metrics: CSV files and plots showing execution times for varying dataset sizes
- Defect Logs: A list of identified bugs and their resolution status
- Final Test Summary: A comprehensive report summarizing test results, coverage, and performance metrics

4.9 Testing Budget

- The testing process will leverage free and open-source tools, ensuring minimal cost.
- Resources needed:
    - Developer time for test creation and execution
    - Computational resources for running performance and mutation tests

- Estimated Budget: ~$0 (using existing hardware/software and open-source tools)

4.10 Tools

- Unit Testing Tools
    - unittest in Python for writing and executing unit tests
- Configuration Management Tools
    - GitHub for version control and collaboration
- Bug Tracking Tools
    - GitHub Issues to track and manage bugs and enhancements

4.11 Automation Strategy

- Automate all unit, integration, and system tests using Python scripts and mocking frameworks
- Use unittest.mock for simulating unavailable components like scanners and databases
- Continuous integration tools (e.g., GitHub Actions) for running tests automatically on each code commit

4.12 Types of Testing, Methodologies, and Techniques

- **Requirement Reviews:** Manual review of specifications to ensure all requirements are clear and testable
- **Design Reviews:** Validate that the system design aligns with ICAO standards and project requirements
- **Unit Testing:** Test each function individually, covering normal, edge, and invalid inputs
- **OATS (Orthogonal Array Testing Strategy):** Apply to test various combinations of MRZ fields for validation
- **Integration Testing:** Validate interactions between components, such as decoding results feeding into validation
- **Usability Testing:** Simulate end-user scenarios to ensure the system is intuitive and produces correct outputs
- **Performance Testing:** Analyze execution time and memory usage for datasets of increasing size (e.g., 100 to 10,000 records)
- **Reliability Testing:** Stress-test the system with continuous inputs to identify any breakdowns or memory leaks
- **System Testing:** Simulate end-to-end scenarios to verify that all components work together correctly
- **Alpha/Beta Tests:** Conduct limited alpha tests internally and share with stakeholders for feedback before release

4.13 Test Platform

- **Environment:** Python 3.7.15 within an Anaconda virtual environment

- **Hardware:** Standard development laptops with at least 8GB RAM and multicore processors
- **Software:**
    - coverage.py for coverage analysis
    - MutPy for mutation testing
    - matplotlib or Excel for generating performance plots

4.14 Progress Measurement

- **Reports and Metrics**
    - **Code Coverage:** Percentage of code covered by tests (target >90%)
    - **Mutation Testing:** Ratio of mutants killed to total mutants generated
    - **Performance Graphs:** Execution times for increasing dataset sizes
- **Sample Reports**
    - **Coverage Report:** Highlighting uncovered lines in the code
    - **Defect Logs:** Tracking bugs and their resolution status
    - **Performance Charts:** Plots of execution time vs. dataset size

4.15 Readiness for System Shipment

- All test cases have been executed with a pass rate exceeding 90%.
- Code coverage and mutation testing results meet or exceed established thresholds.
- Performance metrics are within acceptable limits for all dataset sizes.
- All critical and high-severity bugs have been resolved.
- Approval from all stakeholders, including project manager and QA lead, is secured.

This ensures the system is strong, compliant with international standards, and ready for deployment.

5. Schedule

Start Date	**End Date**	**Phase**	**Description**
Jan 1, 2024	Jan 15, 2024	Requirement Analysis & Planning	Define project requirements and prepare the test plan
Jan 16, 2024	Feb 29, 2024	Unit Testing Development & Execution	Develop unit tests and execute them to validate individual components
Mar 1, 2024	Mar 31, 2024	Integration Testing	Test interactions

			between different modules and components
Apr 1, 2024	Apr 30, 2024	Performance & System Testing	Assess system performance under various conditions and complete end-to-end tests
May 1, 2024	Jul 31, 2024	Bug Fixing & Final Validations	Resolve identified defects and validate all functionalities
Aug 1, 2024	Aug 31, 2024	Test Plan Review & Approval	Final review of the test plan and approval for system delivery
Sep 1, 2024	Sep 1, 2024	Delivery Deadline	Deliver the fully tested and validated system, along with comprehensive documentation and final test report

6. Approvals

Identifying the key roles of stakeholders who should approve the plan is essential to ensure that all aspects of the project align with technical, quality, and business requirements. Stakeholders such as the Project Manager, Development Lead, Quality Assurance Lead, and Client/End-User Representative play critical roles in reviewing and validating the test plan. Their collective approval ensures that the testing approach is comprehensive, the methodologies are effective, and the system meets both functional and user expectations.

Stakeholder Role	**Responsibility**	**Approval Focus**
Project Manager	Oversees the overall project and ensures alignment with goals	Final approval of the test plan and schedule
Development Lead	Ensures technical feasibility and alignment with development processes	Validation of test cases and coverage
Quality Assurance Lead	Verifies the thoroughness and	Approval of testing

	effectiveness of the test plan	methodologies and deliverables
Client/End-User Representative	Ensures the system meets business needs and compliance requirements	Acceptance of system functionalities and outputs
Stakeholder Committee	Provides oversight and ensures adherence to project objectives	Final sign-off on the test plan and readiness for delivery