SurgeVision CMPE 492 - Senior Design Project II

Test Plan Report

TEAM MEMBERS:

HAKAN UCA

ABDULLAH DOĞANAY

DENİZ POLAT

ONUR USLU

Table of Contents

1. Features to Be Tested 3
   1.1 Non-Functional Features 3
2. Testing Methodology 4
   2.1 Unit Testing 4
   2.2 Integration Testing 4
   2.3 System Testing 5
   2.4 Performance Testing 5
   2.5 User Acceptance Testing (UAT) 5
   2.6 Beta Testing 6
3. Test Method 6
   3.1. Unit Testing 6
   3.2. Integration Testing 6
   3.3. System Testing 7
   3.4. Performance Testing 7
   3.5. User Acceptance Testing (UAT) 7
   3.6. Beta Testing 8
4. Test Environment 8
   4.1. Hardware Environment 8
   4.2. Software Environment 8
   4.3. Real-World Simulation 9
5. Test Timeline 9
   5.1. Test Dependencies 10
6. Test Monitoring and Tracking Strategy 10
   6.1. Test Result Tracking 10
   6.2 Defect Management 11
   6.3 Bug Resolution Workflow 11
   6.4 Tools and Platforms 11
7. Roles and Responsibilities 12
   7.1 Test Lead / QA Lead 12
   7.2 QA Engineers / Testers 12
   7.3 Development Team 12
   7.4 Test Approver / Quality Gate 12
8. Risks in the Testing Process 13
9. Example Text Case List 14-15

1. Features to Be Tested

This section identifies the functional components of SurgeVision that will undergo rigorous testing.

Feature	Description
MRI Module	Converts MRI data (.nii) into point cloud (.ply) format.
Scan Module	Acquires 3D scans using ARCore-enabled camera systems.
Simplification	Reduces complexity of point clouds to optimize performance.
Surface Model Module	Generates surface mesh from simplified point clouds.
Registration Module	Aligns and synchronizes MRI and scan datasets using ICP & CPD algorithms.
Visualization Module	Renders the final registered 3D models on AR/VR displays or monitors.
User Interface (UI)	Facilitates interaction between the user and system via gesture or controller.
Tracking Module	Tracks objects and updates point clouds dynamically during simulation.

1. Non-Functional Features

- **Performance under load** (real-time point cloud processing)
- **Responsiveness of AR/VR interaction**
- **System latency and frame rate (target: ≥25 FPS)**
- **Security of data transmission via TLS**
- **System stability under extended sessions**

Each testable feature will be tracked with a **Test ID**, execution result, and tester comments.

2. Testing Methodology

Testing will include a blend of manual and automated approaches to validate functionality, usability, performance, and system integration.

2.1 Unit Testing

- **Scope:** Individual modules such as MRI Converter, PointCloudSimplifier, and ICPAligner.
- **Tools:** Google Test (C++), NUnit (C#)
- **Goal:** Verify method outputs, input validation, and error handling.

Test ID	Component	Description
UT-01	MRI Module	Validate .nii → .ply conversion and exception for invalid files.
UT-02	Simplifier	Ensure accuracy of downsampling and outlier removal.

2.2 Integration Testing

- **Scope:** Module interactions (e.g., MRI → Simplifier → Surface Model).
- **Goal:** Confirm modules integrate seamlessly and pass data correctly.
- **Strategy:** Incremental bottom-up testing.

Test ID	Integration Points	Validation
IT-01	MRI + Scan + Registration	Data alignment within 5mm accuracy.
IT-02	Surface Model + Visualization	Surface models render with no visual distortion.

2.3 System Testing

- **Environment:** Meta Quest 3, Unity-based visualization.
- **Tests:** End-to-end workflows such as “Upload MRI” → “Scan” → “Visualize”.
- **Focus:** Stability, real-time responsiveness, and visual accuracy.

Test ID	Scenario	Expected Outcome
ST-01	Full surgery planning session	Seamless processing and rendering of patient data.

2.4 Performance Testing

- **Metrics:** Load handling, FPS, memory usage.
- **Target:** Minimum 25 FPS under high scan load.
- **Tools:** Unity Profiler, Android Logcat.

Test ID	Metric	Benchmark
PT-01	FPS under full scan load	≥25 FPS
PT-02	Point cloud registration speed	≤2 seconds

2.5 User Acceptance Testing (UAT)

- **Participants:** Medical students, surgeons.
- **Focus:** Usability, clarity, interaction intuitiveness.
- **Tool:** Observation checklist + Likert-scale feedback.

Test ID	Task	Evaluation Criteria
UAT-01	Navigate a surgical scenario	Task completion, ease of use, realism score.

2.6 Beta Testing

- **Scope:** Real-world environment test in lab/hospital.
- **Focus:** Uncovered bugs, performance on medical hardware.
- **Duration:** 3–5 days.

3. Test Method

This section details the procedures and techniques to be employed in verifying the performance, integrity, and functionality of the SurgeVision system. The methods outlined herein are designed specifically to address the critical modules—such as MRI, Scan, Simplification, Surface Modeling, Registration, Visualization, User Interface, and Tracking—and ensure that each delivers robust performance under anticipated conditions.

3.1. Unit Testing

- **Objective:** Validate that each independent module or component performs its designated function correctly.
- **Application:**
    - For instance, the MRI module must accurately convert .nii files to the .ply format, correctly handling invalid inputs by generating appropriate error responses.
    - The Simplification module is assessed on its ability to downsample point clouds effectively and remove outliers without losing essential detail.
- **Tools:** Automated testing frameworks such as Google Test (for C++) or NUnit (for C#) will be implemented to streamline and standardize these tests.

3.2. Integration Testing

- **Objective:** Confirm that individual modules interoperate seamlessly, ensuring data flows correctly between components.
- **Application:**
    - Testing scenarios include verifying the data handoff from the MRI or Scan module to the Registration module, ensuring alignment within predefined tolerances.
    - Verifying that the Surface Modeling and Visualization modules collaborate without introducing visual distortions or data inconsistencies.

3.3. System Testing

- **Objective:** Evaluate the end-to-end functionality of the complete SurgeVision workflow in an environment that closely mirrors the operational setting.

- **Application:**
    - Comprehensive test cases will simulate real-world processes such as “Upload MRI – Perform Scan – Visualize 3D Model,” thereby identifying any functional or integration issues across the entire system.

3.4. Performance Testing

- **Objective:** Measure system responsiveness and resource management under varying loads to ensure operational efficiency meets the defined benchmarks.
- **Application:**
    - The system is expected to maintain a minimum frame rate of 25 FPS during high-load scanning scenarios.
    - Additionally, point cloud registration processes are anticipated to complete within a maximum duration of 2 seconds.
- **Tools:** Performance metrics will be captured using tools like Unity Profiler and Android Logcat.

3.5. User Acceptance Testing (UAT)

- **Objective:** Assess the system’s usability and functional adequacy from an end-user perspective, ensuring that it meets the practical needs and expectations of its target audience (e.g., medical students and surgeons).
- **Application:**
    - Users will be tasked with real-life scenarios, such as navigating a surgical simulation, and will provide feedback via structured surveys and Likert-scale evaluations.

3.6. Beta Testing

- **Objective:** Validate the system's performance in authentic operational environments beyond controlled lab conditions.
- **Application:**
    - The beta phase involves deploying the SurgeVision system to a selected group of users over a span of 3–5 days, during which feedback on performance, usability, and potential issues will be collected comprehensively.

4. Test Environment

The test environment for SurgeVision is designed to replicate both the controlled laboratory settings and the dynamic conditions encountered in actual operational scenarios. This dual-approach ensures that both isolated and integrated performance characteristics of the system are thoroughly validated.

4.1. Hardware Environment

- **AR/VR Devices:**
    - Utilization of Meta Quest 3 headsets for testing the AR/VR visualization component, ensuring that immersive interactions perform as intended under realistic conditions.
- **Mobile Devices:**
    - Deployment on ARCore-enabled devices to conduct thorough testing of the Scan module and the associated user interface.
- **High-Performance Workstations:**
    - Dedicated PCs are used for simulation, development, and automated testing, ensuring that performance metrics such as rendering speeds and data processing are accurately monitored.

4.2. Software Environment

- **Development and Test Platforms:**
    - The primary testing framework is built on Unity (version 2021 or later) to closely mimic production conditions.

- **Test Automation Tools:**
    - Frameworks such as Google Test and NUnit support both unit and integration testing, while performance metrics are continuously monitored via Unity Profiler and Android Logcat.
- **Security and Network Configuration:**
    - The test environment will simulate production-grade security, including TLS encryption for data transmissions, to assess the system's resilience against network-related vulnerabilities.

4.3. Real-World Simulation

- **Laboratory Testing:**
    - Rigorous system and performance tests will be conducted in a controlled lab setting that replicates the end-user hardware and network configurations.
- **Field Testing (Beta):**
    - In collaboration with selected pilot users, the system will undergo beta testing in real-world scenarios—such as hospital or clinical environments—to capture authentic usage feedback and operational challenges.

5. Test Timeline

Date	Test	Description
April 15-19	Unit Test	Testing individual modules (e.g., MRI, Simplifier)
April 22–26	Integration Testing	Testing connections between previously tested units
April 29 – May 3	System Testing	End-to-end workflow: MRI → Scan → Visualize
May 6–10	Performance Testing	FPS, processing time under load
May 13–16,	User Acceptance Testing	Usability testing with medical students and surgeons

Date	Test	Description
May 19–23	Beta Testing	Real-world environment (hospital/lab)
May 27–30	Bug Fixing	Addressing feedback from UAT, beta, performance tests
June 3–6	Final Review & Reporting	Final status tracking and reporting via Jira

5.1. Test Dependencies

Test Type	Dependent on
Integration Testing	Unit testing
System Testing	Integration testing
Simplification	Reduces complexity of point clouds to optimize performance.
Performance Testing	Integration + System Testing
UAT	System Testing + Stability confirmation
Beta Testing	UAT + Performance Testing
Bug Fixing	UAT + Beta + Performance feedback

6. Test Monitoring and Tracking Strategy

6.1. Test Result Tracking

*Jira will be used.

Logging Details:

- Pass/Fail status
- Module tested
- Test environment
- Assigned tester
- Observations and remarks

6.2 Defect Management

Bug Logging: Bugs will automatically be recorded in Jira upon test failures.

Bug Report Includes:

- Description
- Reproduction steps
- Screenshots or log traces (if applicable)

Priority Levels:

- Critical (blocks entire system)
- High (affects key functions)
- Medium (workaround exists)
- Low (minor UI/UX or suggestions)

6.3 Bug Resolution Workflow

1. The issue is assigned to a developer via Jira.
2. Developer implements the fix and deploys the updated build.
3. All developers are capable of resolving any issue, regardless of module ownership. Tasks are distributed based on availability and priority.
4. QA/test team performs Regression Testing to confirm resolution.
5. The issue is then closed in Jira and added to the test report.

6.4 Tools and Platforms

Tool/Platform	**Purpose**
Jira	Bug tracking, test execution tracking
Unity Profiler	Measuring performance (FPS, memory usage)
Android Logcat	Debugging logs during mobile testing
Google Test	Unit testing framework for C++
NUnit	Unit integration tests for C# modules

7. Roles and responsibilities

7.1 Test Lead / QA Lead

- Develop detailed test cases and scripts based on the test strategy.
- Oversee all phases of testing, ensuring comprehensive coverage and adherence to established methodologies.
- Coordinate with team members to integrate testing efforts.
- Manage and track defects using tools such as Jira, ensuring prompt resolution and re-testing.

7.2 QA Engineers / Testers

- Execute test cases across all phases, including unit, integration, system, performance, UAT, and beta testing.
- Accurately log and document defects with supporting evidence (screenshots, logs, etc.).
- Validate fixes and perform regression testing to ensure issues are resolved without affecting other system components.
- Report test results and provide feedback for continuous process improvement.

7.3 Development Team

- Implement fixes for any defects identified during testing.
- Conduct preliminary unit testing to ensure code quality before integration.
- Collaborate closely with the QA engineers to understand defects and confirm that issues have been resolved.

7.4 Test Approver / Quality Gate

- Review the final test reports and verify that all critical test cases have been executed and that defects have been resolved.
- Assess whether the system meets all acceptance criteria and quality benchmarks as defined in the test plan.
- Provide official approval or sign-off on the system before it moves to the next stage or is released to production.

8. Risks in the Testing Process

- ## Time Constrains:

  With a tightly scheduled testing timeline (e.g., specific dates for Unit, Integration, System, Performance, UAT, and Beta testing), any unexpected delays or issues may compromise the entire testing process.

- ## Incomplete Documentation:

  Insufficient or outdated documentation on system components can hinder the creation of comprehensive test scenarios, leading to gaps or misunderstandings in the test coverage.

- ## Hardware Issues:

  Problems with the test environment's hardware—such as AR/VR devices (e.g., Meta Quest 3), ARCore-enabled mobile devices, and high-performance PCs—might affect the accuracy and consistency of test results.

- ## Delayed User Feedback:

  In both UAT and Beta testing phases, if user feedback is not received on time, identifying and resolving issues may be delayed, impacting the overall test reporting.

- ## Software Integration Issues:

  Integration challenges between different modules (e.g., MRI, Scan, Simplification, Registration, Surface Modeling, and Visualization) could lead to data transfer inconsistencies or failures in module interactions.

- ## Performance Problems:

  The system might underperform under heavy loads, not meeting the specified criteria (e.g., maintaining at least 25 FPS and completing point cloud registration within 2 seconds), which would negatively affect the user experience and system reliability.

9. Example Test Case List

TC-001: MRI Module Conversion Test

- **Test ID:** TC-001
- **Feature to be Tested**: The functionality of the MRI module in converting .nii files into .ply format.
- **Input Data:** A valid, error-free MRI .nii file.
- **Expected Output:** The .nii file is correctly converted into a .ply file without generating any error messages.

TC-002: MRI Module Invalid File Handling

- **Test ID:** TC-002
- **Feature to be Tested:** How the MRI module handles invalid file formats.
- **Input Data:** An improperly formatted file (e.g., a .txt file).
- **Expected Output:** An appropriate error message is produced, indicating that the conversion has failed.

TC-003: Integration Test – MRI and Registration Modules

- **Test ID:** TC-003
- **Feature to be Tested:** The transfer of data from the MRI module to the Registration module, with an alignment accuracy within 5mm.
- **Input Data:** A valid MRI data set along with proper registration parameters.
- **Expected Output:** The data is transferred seamlessly with the registration achieving an accuracy within the specified 5mm range.

TC-004: Performance Test – FPS under Load

- **Test ID:** TC-004
- **Feature to be Tested:** The system's FPS performance in an AR/VR setting under heavy load.
- **Input Data:** A scenario with a high volume of scanned data (e.g., a densely populated 3D data set).
- **Expected Output:** The system maintains a minimum FPS of 25 even under heavy data load.

TC-005: User Acceptance Testing (UAT) – UI Interaction

- **Test ID:** TC-005
- **Feature to be Tested:** The usability and intuitive behavior of the User Interface (UI).
- **Input Data:** A real-life scenario where a user uploads patient data, performs a scan, and views the 3D model.
- **Expected Output:** The user should be able to easily navigate through the process with clear outputs and high satisfaction as evidenced by positive feedback on the survey.