**Test Plan:**

Cruise Liner Management Services

**Prepared by:**

Group 3

9/3/2023

# TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 OBJECTIVES AND TASKS

2.1 Objectives

2.2 Tasks

3.0 SCOPE

4.0 Testing Strategy

4.1 Alpha Testing (Unit Testing)

4.2 System and Integration Testing

4.3 Performance and Stress Testing

4.4 User Acceptance Testing

4.5 Batch Testing

4.6 Automated Regression Testing

4.7 Beta Testing

4.8 Manual Regression Testing

5.0 Hardware Requirements

6.0 Environment Requirements

6.1 Main Frame

6.2 Workstation

7.0 Test Schedule

8.0 Control Procedures

9.0 Features to Be Tested

10.0 Features Not to Be Tested

11.0 Resources/Roles & Responsibilities

12.0 Schedules

13.0 Significantly Impacted Departments (SIDs)

14.0 Dependencies

15.0 Risks/Assumptions

16.0 Tools

17.0 Approvals

**1.0 INTRODUCTION**

The Cruise Liner Management Services (CLMS) is a sophisticated application designed to bring efficiency, speed, and user-friendliness to the cruise liner industry. This software seeks to address multiple pain points in the industry such as inefficient tracking of ships, complex reservation systems, cumbersome billing processes, and suboptimal maintenance schedules.

**2.0 OBJECTIVES AND TASKS**

# 2.1 Objectives

The Master Test Plan (MTP) serves as the blueprint for all testing activities. It outlines the strategy, scope, resources, and schedule for the intended testing activities to ensure that CLMS meets its design specifications and requirements.

**2.2 Tasks**

1. Preparation of Test Environment
   * Setting up the software and hardware needed for the test environment.
2. Requirement Validation
   * Verify that all requirements from the project scope are represented in the test plan.
3. Test Case Design
   * Develop test cases to cover each of the functionalities mentioned in the project scope.
4. Unit Testing
   * Testing the smallest parts of the application in isolation from each other.
5. System and Integration Testing
   * Ensuring that the whole system works well together, especially where different units/modules intersect.
6. Performance and Stress Testing
   * Verifying that the system can handle the maximum workload and testing its behavior under extreme conditions.
7. User Acceptance Testing
   * Validate that the system's functionality aligns well with user requirements and is ready for operational use.
8. Automated Regression Testing
   * Ensuring that new changes haven't broken any existing functionalities.
9. Problem Reporting
   * Logging any issues or bugs found during the testing phase and tracking their resolution.
10. Post-Testing Activities
    * Includes the cleaning up of test environments and the consolidation of test and defect reports.
11. Documentation
    * Complete all documentation required for the testing, including the preparation of a final test summary report.
12. Presentation
    * Prepare for the presentation of testing results to classmates and the professor.

**3.0 SCOPE**

# General

The scope of this Master Test Plan (MTP) covers multiple dimensions of the Cruise Liner Management Services (CLMS) software:

* **Functional Aspects**:
  + Testing the core functionalities of the software like real-time ship tracking, guest reservations, billing, and maintenance scheduling.
* **Existing Interfaces**:
  + Verifying that the CLMS interfaces with existing databases, third-party services, and other systems as specified in the requirements.
* **Integration**:
  + Ensuring seamless integration of the software's different modules, such as the user interface, back-end services, and databases.
* **User Acceptance Testing (UAT)**:
  + Ensuring the software meets the expectations and requirements of the end-users, including ease of use, performance, and reliability.
* **Performance & Scalability**:
  + Evaluating how the application performs under different conditions, including high server load and large database queries.
* **Security**:
  + Assessing the security features to ensure data integrity, confidentiality, and availability.

# Tactics

To accomplish the testing items listed in the scope, the following tactics will be employed:

* **Functional Testing**:
  + The team will use automated tests where applicable and manual tests for specific use cases that can't be automated.
* **Interface Testing**:
  + The User Experience (UX) designer will work closely with end-users to gather their thoughts of the application. These findings will be shared with the team to ensure they are implemented.
* **Integration Testing**:
  + After individual module tests are successful, an end-to-end integration test will be conducted to ensure seamless interaction between all modules.
* **User Acceptance Testing (UAT)**:
  + Select end-users will be invited to participate in UAT. Feedback will be collected and analyzed to make necessary adjustments.
* **Performance & Scalability**:
  + The testing of performance and scalability will be a continuous process throughout the development. This will be executed by an automated application.
* **Security**:
  + All members of the team will be responsible for testing the overall security of the application. If any security risks are identified, the developers will be notified and make the necessary adjustments.

# 4.0 TESTING STRATEGY

* **Track Current and Projected Locations of All Ships**
  + **Approach**: Automated GPS simulation tests and real-world tracking tests
  + **Major Activities**: Data accuracy verification, system reliability
  + **Tools**: Web browser, internet access, CLMS application and database
* **Track Port Departures and Arrivals**
  + **Approach**: Database consistency tests, and manual validation
  + **Major Activities**: Input validation, port entry and exit logs, time-stamp verification
  + **Tools**: CLMS application and database
* **Guest Reservations and Reservation Management**
  + **Approach**: Database integrity checks and User Interface (UI) testing
  + **Major Activities**: Booking flow, database consistency, and guest data validation
  + **Tools**: CLMS application and database
* **Billing and Payment Status**
  + **Approach**: Automated transaction simulations, manual review
  + **Major Activities**: Payment gateway tests, currency conversion, and invoice generation
* **Tools**: CLMS application and database
* **Occupancy and Capacity Statistics (Current and Historical)**
  + **Approach**: Database query tests, manual validation
  + **Major Activities**: Data consistency checks, historical data validation
  + **Tools**: CLMS application and database
* **Manifest Demographical Statistics (Current and Historical)**
  + **Approach**: Data analytics tests
  + **Major Activities**: Data integrity and consistency checks for demographical statistics
  + **Tools**: CLMS application and database
* **Maintenance Service Scheduling and Statistics**
  + **Approach**: Time-based automated and manual tests
  + **Major Activities**: Trigger-based tests for maintenance alerts, schedule conflict checks
  + **Tools**: CLMS application and database

# 4.1 Unit Testing

**Definition:**

The goal of unit testing is to ensure that individual components of the CLMS system are tested in isolation to verify their operational correctness. The minimum degree of comprehensiveness is to cover at least 90% of code lines and 85% of branches. Test coverage tools will be used to measure this comprehensiveness.

**Participants:**

**Software Designers**: Daniel Cloonan & Anthony Raarup

**Test Director**: Ti-Andre Sabur

**Requirements Manager**: Chyna Pritchett

**Methodology:**

1. **Test Script Writing**:
   * Test scripts will be written by the Test Director in collaboration with the Software Designers. These scripts will cover all edge cases, boundary conditions, and typical scenarios.
2. **Sequence of Events**:
   1. Initial Draft of Test Scripts
   2. Review by Software Designers and Requirements Manager
   3. Approval by Project Manager
   4. Execution of Unit Tests
   5. Analysis of Test Results
   6. Code Refinement (if necessary)
   7. Final Run of Unit Tests
   8. Documentation of Results

# 4.2 System and Integration Testing

**Definition:**

System and Integration Testing for the Cruise Liner Management Services (CLMS) involves validating that the different modules or components of the system work together seamlessly and that the entire system behaves as expected. This will encompass the features outlined in the project scope.

**Participants:**

**Project Manager**: Neon Sapkota (Overseeing the process)

**Test Director**: Ti-Andre Sabur (Lead)

**Software Designers**: Daniel Cloonan & Anthony Raarup

**User Experience/Training Manager**: Chyna Pritchett

**Requirements Manager**: Chyna Pritchett (For ensuring requirements match tests)

**Methodology:**

1. **Test Script Writing**:
   * The Test Director will create the System and Integration Test scripts. These will be reviewed by the Software Designers, Project Manager, and Requirements Manager to ensure alignment with the project scope.
2. **Sequence of Events**:
   1. Drafting of Test Scripts for System and Integration Testing
   2. Internal review by Test Director, Software Designers, and Requirements Manager
   3. Approval by Project Manager
   4. Pre-test Environment Setup
   5. Execution of System and Integration Tests
   6. Compilation and Analysis of Test Results
   7. If failures occur, feedback to Software Designers for remediation
   8. Re-testing after modifications (if needed)
   9. Final Documentation and sign-off

# 4.3 Performance and Stress Testing

**Definition:**

Performance and Stress Testing for the Cruise Liner Management Services (CLMS) project involves assessing how the system performs under various conditions, including high user loads, large amounts of data, and extreme operational demands. This type of testing is crucial to determine the system's reliability, robustness, and resilience under stress. The features to be stress-tested will correspond with the scope of the project, particularly focusing on real-time tracking, reservation management, billing automation, and statistical analysis capabilities.

**Participants:**

**Project Manager**: Neon Sapkota (Overseeing the process)

**Test Director**: Ti-Andre Sabur (Lead)

**Software Designers**: Daniel Cloonan & Anthony Raarup

**User Experience/Training Manager**: Chyna Pritchett

**Requirements Manager**: Chyna Pritchett (For ensuring requirements match tests)

**Methodology:**

1. **Test Script Writing**:
   * The Test Director will create Performance and Stress Test scripts.
   * These will undergo review by the Software Designers and Requirements Manager to align with project scope and will require final approval from the Project Manager.
2. **Sequence of Events**:
   1. Development of Performance and Stress Test Scripts
   2. Internal review of scripts for alignment with project scope
   3. Approval from Project Manager
   4. Set-up of Testing Environment mimicking extreme operational scenarios
   5. Execution of Performance and Stress Tests
   6. Analysis of Test Results:
      * (CPU usage, memory usage, latency, throughput, error rates, etc.)
   7. Identification and reporting of bottlenecks or issues
   8. Code and System Optimization based on Test Results
   9. Re-testing after modifications (if needed)
   10. Final Documentation and sign-off f

# 4.4 User Acceptance Testing

**Definition:**

User Acceptance Testing (UAT) is the final phase of the testing process, performed to verify that the Cruise Liner Management Services (CLMS) meets its initial requirements and is ready for deployment. In this phase, the end-users (i.e., fleet managers, reservation staff, billing clerks, etc.) evaluate the system to ensure it aligns with the scope and requirements for operational use.

**Participants:**

**Project Manager**: Neon Sapkota (Overseeing the process)

**User Experience/Training Manager**: Chyna Pritchett (Lead and point of contact with end-users)

**Test Director**: Ti-Andre Sabur (Supporting role, ensuring test case comprehensiveness)

**Requirements Manager**: Chyna Pritchett (Ensuring that test cases align with initial requirements)

**End-Users**: Fleet managers, reservation staff, billing clerks (Selected representatives for actual testing)

**Methodology:**

1. **Test Script Writing**:
   * The User Experience/Training Manager will collaborate with the Requirements Manager and Test Director to create UAT scripts.
   * These scripts will then undergo a review for alignment with the project's scope and will require final approval from the Project Manager.
2. **Sequence of Events**:
   1. Development of UAT Test Scripts
   2. Internal review of scripts for alignment with initial requirements
   3. Approval from Project Manager
   4. Training Session for End-Users on how to perform UAT
   5. Execution of User Acceptance Tests
   6. Collection of User Feedback
   7. Analysis of Test Results and Feedback
   8. Identification and reporting of any issues or discrepancies
   9. Implementation of Necessary Fixes
   10. Re-testing (if required)
   11. Final Documentation and sign-off

**4.5 Batch Testing**

**Definition:** Batch Testing is a testing phase where a set of test cases are executed as a group or batch. It is typically used to verify the functionality of the system when multiple related test cases need to be executed together to simulate real-world scenarios.

**Participants:**

**Project Manager**: Neon Sapkota (Overseeing the process)

**User Experience/Training Manager**: Chyna Pritchett (Lead and point of contact with end-users)

**Test Director**: Ti-Andre Sabur (Supporting role, ensuring test case comprehensiveness)

**Requirements Manager**: Chyna Pritchett (Ensuring that test cases align with initial requirements)

**End-Users**: Fleet managers, reservation staff, billing clerks (Selected representatives for actual testing)

**Methodology:**

* Test Case Selection: The Test Analysts will select a group of test cases that need to be executed together in a batch. These test cases are chosen based on their interdependencies or relevance to a specific scenario.
* Batch Creation: The Test Analysts will create a batch of selected test cases, defining the order of execution and any prerequisites.
* Test Environment Setup: Ensure that the test environment is properly configured to support batch testing.
* Batch Execution: Execute the batch of test cases as per the defined order, following the established test scripts and documenting the results.
* Defect Identification: If any defects or issues are identified during batch testing, they are reported to the development team for resolution.
* Regression Testing: After defect resolution, re-run the batch testing to ensure that the changes do not introduce new issues.
* Reporting: Document the results of batch testing, including any defects found and their status.

**4.6 Automated Regression Testing**

**Definition:** Automated Regression Testing is the process of using automated testing tools to retest the existing functionality of a software application to ensure that new code changes or modifications have not introduced unintended side effects or broken existing features.

**Participants:**

**Project Manager**: Neon Sapkota (Overseeing the process)

**User Experience/Training Manager**: Chyna Pritchett (Lead and point of contact with end-users)

**Test Director**: Ti-Andre Sabur (Supporting role, ensuring test case comprehensiveness)

**Requirements Manager**: Chyna Pritchett (Ensuring that test cases align with initial requirements)

**End-Users**: Fleet managers, reservation staff, billing clerks (Selected representatives for actual testing)

**Methodology:**

* Test Suite Selection: Identify the critical test cases that need to be automated for regression testing. These are typically core functionalities that should not break with code changes.
* Test Automation Tool Selection: Choose appropriate test automation tools and frameworks for creating and executing regression test scripts.
* Test Script Development: Test Automation Engineers create and validate automated test scripts based on the selected test cases.
* Integration with Continuous Integration (CI) System: Integrate the automated regression test suite with the CI system to run tests automatically whenever new code is pushed to the repository.
* Scheduled Execution: Set up a schedule for automated regression test suite execution, which can be daily, after each code commit, or as needed.
* Result Analysis: Review automated test results to identify any failures or anomalies.
* Defect Reporting: If test failures occur, defects are reported to the development team for resolution.
* Maintenance: Periodically update the automated regression test suite to accommodate changes in the application's functionality or user interface.

**4.7 Beta Testing** **Participants:**

Beta testing involves real end-users or external stakeholders who voluntarily participate in testing a pre-release version of the software. The selection of beta testing participants will depend on the goals and scope of the testing effort. Participants may include:

* End-users: These are the individuals who will ultimately use the software. They provide valuable feedback on usability, user experience, and functionality.
* Customers: If applicable, existing customers or clients of the software may be invited to participate in beta testing to gather their insights and ensure that the software meets their needs.
* External stakeholders: This could include partners, vendors, or third-party organizations that have a vested interest in the software's performance and functionality.
* Selected target audience: Depending on the software's purpose, a specific group of individuals may be chosen to participate in beta testing. For example, if the software is intended for healthcare professionals, healthcare practitioners may be invited.
* Diverse user groups: It's often beneficial to include a diverse set of users to ensure that the software caters to a wide range of needs and scenarios.

The goal of beta testing is to gather real-world feedback and uncover any issues that may not have been identified during earlier testing phases. Beta testers play a crucial role in validating the software's readiness for a wider audience and helping developers make necessary improvements before the official release.

**4.8 Manual Regression Testing**

Definition: Manual regression testing, within the context of Automated Regression Testing, refers to the human validation and review of test results generated by automated test scripts to ensure that the application's existing functionality remains unaffected after code changes or modifications.

Participants:

* Project Manager: Neon Sapkota (Overseeing the process)
* User Experience/Training Manager: Chyna Pritchett (Lead and point of contact with end-users)
* Test Director: Ti-Andre Sabur (Supporting role, ensuring test case comprehensiveness)
* Requirements Manager: Chyna Pritchett (Ensuring that test cases align with initial requirements)
* End-Users: Fleet managers, reservation staff, billing clerks (Selected representatives for actual testing)
* Manual Testers: Skilled testing professionals responsible for reviewing automated test results.

Methodology:

* Test Suite Selection:
  + Identify the critical test cases that need manual verification as part of the regression testing process.
  + Focus on high-priority test cases and those related to core functionalities.
* Manual Verification:
  + Skilled manual testers review the automated test results to identify any failures or anomalies.
  + Manual testers use their expertise to validate if the application behaves correctly post-code changes.
* Defect Reporting:
  + If manual testers discover any discrepancies or issues in the application's behavior, they report these defects.
  + Defects are documented and communicated to the development team for resolution.
* Communication and Collaboration:
  + Manual testers collaborate with the development team to ensure that defects are understood and addressed.
* Documentation:
  + All manual regression testing activities, including test results, defects, and collaboration notes, are documented for reference.
* Feedback Loop:
  + Continuous feedback is provided to the automated testing team to improve the effectiveness of automated test scripts.

Incorporating manual regression testing within Automated Regression Testing ensures a human perspective on the application's behavior, complementing automated checks. This combination helps identify subtle issues that automated tests may overlook, resulting in a more robust and reliable testing process.

# 5.0 HARDWARE REQUIREMENTS

Computers

Modems

**6.0 ENVIRONMENT REQUIREMENTS**

# 6.1 Main Frame

**Necessary Properties**:

* **Hardware**:
  + Personal computer with minimum system requirements to run the Cruise Liner Management Services (CLMS) application.
* **System Software**:
  + Operating system compatible with the CLMS, typically Windows or macOS.
  + Basic database software for local testing, such as SQLite.
* **Communications Software**:
  + Internet connection for communication within the team and for accessing any online resources.
* **Mode of Usage**:
  + The application should be run in the same mode as it would in production, as a standalone application on a personal computer.
* **Security**:
  + Basic firewall and antivirus software to protect the test environment.

**Desired Properties**:

**6.2 Workstation**

# 7.0 TEST SCHEDULE

* Ensuring the database and application are compatible.
* Importing database records into the application upon completion of modules.
* Test each feature (tracking, guest reservations, billing and payment, occupancy and statistics, manifest and maintenance).

**8.0 CONTROL PROCEDURES**

# Problem Reporting

In the case that an incident is encountered, Appendix A will need to be completed. This form will then be emailed to every member of the team. The Project Manager will follow up with the incident within a 24-hour period.

# Change Requests

Any modifications made to the software that is outside of previously discussed plans will require a written request. This request will be given to the Project Manager who will then meet with the developers and requirements manager to ensure the changes are appropriate. These changes must be approved by the project manager and both developers.

**9.0 FEATURES TO BE TESTED**

* **Track Current and Projected Locations of All Ships**
* **Track Port Departures and Arrivals**
* **Guest Reservations and Reservation Management**
* **Billing and Payment Status**
* **Occupancy and Capacity Statistics (Current and Historical)**
* **Manifest Demographical Statistics (Current and Historical)**
* **Maintenance Service Scheduling and Statistics**

# 10.0 FEATURES NOT TO BE TESTED

Each feature will be tested out of a sample set from the database.

# 11.0 RESOURCES/ROLES & RESPONSIBILITIES

The software designers will test the compatibility. Testing directors will test the functionality of the application features.

**12.0 SCHEDULES**

# Major Deliverables

- Test Plan

* Phase 1 test code testing
* Phase 2 test code testing
* Phase 3 test code testing
* Final testing
* Test Incident Reports
* Test Summary Reports

**13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)**

Developers and Testers

# 14.0 DEPENDENCIES

Each testing phase will be dependent on the source code as well as availability of internet.

# 15.0 RISKS/ASSUMPTIONS

Uncontrolled access to the application could lead to an increased risk of piracy.

Contingency plan: Tracking uncontrolled access to the application and notifying the proper authority.

**16.0 TOOLS**

Automation: JUnit testing

Bug tracking: Eclipse

# 17.0 APPROVALS

Group 3 majority vote.

Appendix A

Incident Encountered Report

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Specific Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOTE: You must submit this form to all members of the team. The project manager will follow up within 24-hours. Thank you.