|  |
| --- |
|  |
| ATMP |

| RELATED ARTIFACTS | |  |
| --- | --- | --- |
| Ref. | Artifact Name | Description |
|  | Test strategy | The overall strategy for how the automated tests will be designed and implemented, including the types of testing being performed (such as functional, compatibility, and exploratory testing), as well as any prioritization or risk assessment. |
|  | Test cases | A list of all the test cases that are being automated, including both positive and negative scenarios. |
|  | Test data | Any required input data that is used to execute the automated tests, including test user accounts, restaurant and bar locations, and vaccination and test records. |
|  | Test scripts | The code used to automate the test cases, including scripts for both Android and iOS platforms. |
|  | Test results | The results obtained from executing the automated tests, including screenshots and logs for both successful and failed test cases. |
|  | Test environment setup | Information on how the test environment is set up, including any necessary hardware, software and network configurations. |
|  | Test execution log | A log of all the tests that have been executed and their results, including any necessary metadata such as test duration and device information. |
|  | Defect log | A log of any defects found during testing, including their severity and any necessary steps for replication. |

|  |  |
| --- | --- |
| Acronyms & Abbreviations | |
| SUT | System Under Test |
| GUI | Graphical User Interface |
| API | Application Programming Interface |
| iOS | iPhone Operating System |
| Android | Mobile Operating System for Android |
| UI | User Interface |
| UX | User Experience |
| QA | Quality Assurance |
| TC | Test Case |
| TR | Test Report |
| TA | Test Automation |
| CI/CD | Continuous Integration/Continuous Deployment |
| ROI | Return on Investment |
| KPI | Key Performance Indicator |

Contents

[1 Project Specific Impact to Testing 4](#_Toc132814448)

[2 Scope of Testing 4](#_Toc132814449)

[2.1 In Scope 4](#_Toc132814450)

[3 Quality and Acceptance Criteria 5](#_Toc132814451)

[4 Test Process Description 5](#_Toc132814452)

[4.1 Test Planning Phase 5](#_Toc132814453)

[4.1.1 Entry/Exit Criteria for Each Phase 5](#_Toc132814454)

[4.1.2 Defects Severity Description 7](#_Toc132814455)

[4.2 Test Design Phase 7](#_Toc132814456)

[4.2.1 Test Cases Creation Rules 7](#_Toc132814457)

[4.2.2 Managing Test Cases in the Test Management Tool 8](#_Toc132814458)

[4.3 Test Execution Phase 8](#_Toc132814459)

[4.3.1 Defect Reports Raising and Management Rules 8](#_Toc132814460)

[4.3.2 Defect Lifecycle 9](#_Toc132814461)

[4.4 Test Reporting Phase 9](#_Toc132814462)

[4.4.1 Test Report Containment 9](#_Toc132814463)

[4.4.2 List of the Metrics to Track on a Project 10](#_Toc132814464)

[5 Risk Management 11](#_Toc132814465)

[6 Test Team 12](#_Toc132814466)

[6.1 Roles and Responsibilities 12](#_Toc132814467)

[6.2 Stakeholders 13](#_Toc132814468)

[6.3 Communication Plan 13](#_Toc132814469)

[6.4 Escalation Plan 14](#_Toc132814470)

[7 Test Schedule 14](#_Toc132814471)

[8 Test Deliverables 15](#_Toc132814472)

# Project Specific Impact to Testing

The project is a mobile application designed to help users shape their social life by discovering restaurants, bars, and cafes, accessing menus and other information, and managing reservations and payment receipts. This will influence the testing process as the application will need to be thoroughly tested for its functional capabilities, compatibility with different devices and platforms, localization, usability, and other factors. The organization's specific constraints and assumptions will also affect the testing process, as well as any critical success factors that need to be considered. Finally, the project geography, with its location in Germany, may have an impact on the testing process due to time zone differences and language requirements.

Project phase: Development phase.

Technology stack: Java, Swift and JS.

Constrains and assumptions:

* SUT is a mobile application, which requires different approaches in testing, as compared to web or desktop applications.
* The application involves payment processing and storage of sensitive user data, which requires special attention to security testing.
* There may be changes in the technology stack during the development phase, which will require changes in the test approach and types.
* Mobile devices have limited resources, which may affect the test execution and types.

Critical success factors:

* Ensuring proper test coverage of all the features and functionalities of the application.
* Ensuring the security and confidentiality of user data.
* Ensuring that the application meets the performance and usability requirements.
* Ensuring that the testing is aligned with the development process and is performed in a timely manner.

Project geography: The project is based in Europe, and the testing team is expected to work in the same time zone as the development team to ensure timely communication and collaboration.

# Scope of Testing

The project consists of four applications:

1. iOS mobile application for end-users
2. Android mobile application for end-users
3. Web application for location staff
4. Mobile application for location staff

Each application serves a different purpose. The iOS and Android mobile applications are for end-users to digitally shape their social life. The web application is for location staff to provide information about locations, create menus, check bills, and view transaction overviews. The mobile application for location staff serves the same purpose as the web application, but in a mobile format.

## In Scope

|  |  |  |  |
| --- | --- | --- | --- |
| System/component/interface under test | Description | Responsible side | Reference |
| Mobile App - iOS | Mobile application for end-users on iOS platform | QA Team | iOS App Specification document, version 1.0 |
| Mobile App - Android | Mobile application for end-users on Android platform | QA Team | Android App Specification document, version 1.0 |
| Web App | Web application for location staff to manage information about location, menus, transactions | QA Team | Web App Specification document, version 1.0 |
| Mobile App - iOS for Location Staff | Mobile application for location staff on iOS platform | QA Team | iOS App Specification document for Location Staff, version 1.0 |
| Mobile App - Android for Location Staff | Mobile application for location staff on Android platform | QA Team | Android App Specification document for Location Staff, version 1.0 |

# Quality and Acceptance Criteria

1. All critical and high priority defects identified during testing must be fixed and retested before the application can be released.
2. The automation tests should cover at least 70% of the functional requirements.
3. The manual testing should cover the remaining 30% of the functional requirements and all non-functional requirements such as performance and security.
4. The application should pass the regression testing after each sprint and before release.
5. All security and accessibility testing must be completed and passed before release.
6. The application should be localized and tested for different languages including English and German.

# Test Process Description

## Test Planning Phase

### Entry/Exit Criteria for Each Phase

**1. Test Planning Phase:**

Entry Criteria:

* Test environment is set up and available for testing
* Test data is available for testing
* Test resources are identified and allocated
* Test schedule is agreed upon and communicated to all stakeholders
* Test tools are identified and available for use
* Risks and issues are identified and documented

Exit Criteria:

* All test scenarios and cases are identified and documented
* Test estimation and resource allocation is done
* Test schedule is finalized
* Test readiness review is conducted and approved
* Test environment and data are verified and validated
* Test tools are tested and verified
* Risks and issues are mitigated or addressed

**2. Test Design Phase:**

Entry Criteria:

* Requirements are available and clear
* Test environment is set up and available for testing
* Test data is available for testing
* Test design techniques are chosen and applied
* Test coverage is defined and documented

Exit Criteria:

* All test cases are identified and documented
* Test cases are reviewed and approved by Team Lead
* Test coverage is defined and documented
* Test cases are mapped to requirements
* Traceability matrix is prepared
* Test environment and data are verified and validated

**3. Test Execution Phase:**

Entry Criteria:

* Test cases are identified and documented
* Test cases are reviewed and approved
* Test data is available for testing
* Test environment is set up and available for testing
* Test scripts are created and reviewed
* Test schedule is communicated to all stakeholders
* Defect management process is defined and communicated
* Test progress is tracked and reported

Exit Criteria:

* All test cases are executed and documented
* Test execution results are reviewed and approved
* Defects are identified, tracked, and documented
* Defects are triaged and prioritized
* Defects are resolved and verified
* Test coverage is measured and reported
* Test environment and data are released
* Test completion report is prepared and reviewed

**4. Reporting Phase:**

Entry Criteria:

* All test cases are executed and documented
* Defects are identified, tracked, and documented
* Test coverage is measured and reported
* Test completion report is prepared and reviewed

Exit Criteria:

* Test summary report is prepared and reviewed
* Test metrics are analysed and reported
* Test environment is decommissioned

### Defects Severity Description

|  |  |  |
| --- | --- | --- |
| Severity | Meaning | Examples |
| Blocker | Behavior causes the business/consumer to stop working. Application under test crashes or inoperable | App crashes on startup, app cannot connect to the server. |
| Critical | Behavior causes the business/consumer to operate in a limited way. No workaround available. | App freezes when user tries to make a payment, incorrect information is displayed on the screen. |
| Major | Behavior allows business/consumer to continue working in a limited way with work around. | App displays incorrect data in a transaction details, but user can still access the correct data through transactions overview. |
| Minor | Behavior has no functional impact on business/consumer and can also be attributed to UI aesthetics. | A typo in a label or a minor formatting issue. |
| Trivial | The defect does not affect functionality or data, it is merely an inconvenience. | A cosmetic issue like misaligned text or a slightly off-color button. |

## Test Design Phase

### Test Cases Creation Rules

Test Case Format: Test cases should be created and managed in TestRail using the predefined format. The format should include a unique identifier, test case name, preconditions, test steps and expected results. Test cases should be written in clear and concise language, free from any ambiguity, and should be easy to understand.

Test Steps Format: Test steps should include an action verb that describes the action to be performed, a description of the input or data required, and expected output. Test steps should be simple and atomic, covering one action at a time. Test data should be clearly defined and documented, including valid and invalid inputs to test the application under different scenarios.

Test Data Preparation and Management: Test data should be prepared and stored in “Preconditions” field in TestRail for each test case. Test data should be diverse, covering a wide range of scenarios, including edge cases, boundary values, and negative scenarios. Test data should be well-documented, and any changes to the test data should be tracked and documented in TestRail.

Test Case Review: All test cases should be reviewed by the team lead or senior members of the team in TestRail. The review should ensure that the test cases cover all possible scenarios and follow the defined format, steps format, and guidelines. Reviewers should provide constructive feedback and suggestions for improvement in comments section in TestRail. Any issues or gaps identified during the review should be documented in TestRail, and the necessary corrections should be made before moving to the next phase.

Test Coverage: Test coverage should be defined for each requirement or user story. The team should ensure that all requirements or user stories have been covered by the test cases. Test coverage should include functional, non-functional, and edge cases, and should be reviewed regularly. Any gaps in the test coverage should be addressed promptly to ensure comprehensive testing.

Automation Test Case Creation: Test cases selected for automation should be created in TestRail using the predefined format and should include all necessary information, such as input data, expected results, and assertions. Automation scripts should follow established coding guidelines and best practices, including modularization, error handling, and reusability. Automation scripts should be reviewed by the automation team for correctness and efficiency and linked to the corresponding test cases in TestRail.

Manual Test Case Creation: Manual test cases should be created in TestRail using the predefined format and should include all necessary information, such as input data, expected results, and steps to reproduce. Manual test cases should be reviewed by the testing team for clarity and accuracy. Manual testing should be performed following established test procedures and guidelines, and any defects found should be logged, documented in Jira and linked to the corresponding test case.

### Managing Test Cases in the Test Management Tool

TestRail is going to be used as the test management tool. The tool allows teams to create, organize, and execute test cases, track testing progress, and generate reports. This guide will provide instructions on managing test cases in TestRail.

Organizing Test Suites: In TestRail, test cases are organized into test suites. Test suites are used to group related test cases together. It is important to organize test suites in a logical and easy-to-understand manner, such as by feature or functionality.

Creating Test Cases: To create a test case, navigate to the appropriate test suite and click the “Add Test Case” button. Enter a name, description, preconditions and any necessary steps and expected results. You can also add attachments, labels, and custom fields as needed.

Running a Test Cycle: To run a test cycle, create a new test run within the appropriate test suite. You can select which test cases to include in the test run and assign them to testers. Testers can then execute the test cases and record the results in TestRail. TestRail allows you to track the progress of the test run and generate reports.

Recording Test Results: After executing a test case, testers should record the results in TestRail. TestRail provides a pass/fail option, as well as the ability to record comments, defects, and screenshots. Testers can also assign defects to team members for further investigation and resolution.

Reporting: TestRail provides various reporting options to track testing progress, identify trends, and monitor quality metrics. Reports can be generated for individual test cases, test runs, and test suites.

## Test Execution Phase

### Defect Reports Raising and Management Rules

When raising defects, it is expected from the team to provide a clear and concise description of the issue, along with any relevant screenshots, logs, or other supporting information. The defect report should include the following fields:

Summary: A brief summary of the issue, should answer questions “What?”, “Where?”, “When?”.

Description: A detailed description of the issue, including preconditions, steps to reproduce, expected behaviour, and actual behaviour.

Severity: The severity of the issue, based on its impact on the application and users.

Environment: The environment in which the issue was encountered, including device, OS, and browser details.

Attachments: Any relevant screenshots, logs, or other supporting information.

By default, the appropriate component Product Owner should be assigned to the defect. The team lead should review the defect report before assigning it to the appropriate component Product Owner. The review process should ensure that the defect report is complete, accurate, and follows the defined format and guidelines. Any issues or gaps identified during the review should be documented, and the necessary corrections should be made before assigning the defect to the appropriate component Product Owner.

### Defect Lifecycle

The rules for each status change are as follows:

* Created - This is the initial status of a bug when it is first reported.
* Ready for Dev - This status is assigned when the team lead has reviewed the bug and determined that it is valid and needs to be fixed.
* In Progress - This status is assigned when the developer starts working on the bug.
* Review - This status is assigned when the developer has completed work on the bug and it is ready for code review.
* Ready - This status is assigned when the code review process is complete and the bug is ready for testing.
* On verification - This status is assigned when the bug has been deployed on the QA environment and is being tested by the assigned tester.
* Reopened - This status is assigned if the tester finds issues with the bug during testing and the bug is not fixed.
* Done - This status is assigned when the bug has been fixed and verified by the QA Engineer.

Once a bug is marked as "Done", the resolution field must be set. The available resolutions in JIRA are:

* Fixed - This resolution is used when the bug has been fixed by the developer and verified by the QA Engineer.
* Duplicate - This resolution is used when the bug report is a duplicate of another issue that is already in the system.
* Cannot Reproduce - This resolution is used when the bug is not reproducible anymore.

The resolution chosen depends on the nature of the bug and the team's decision on how to handle it. For example, if the bug has been fixed and verified, the resolution should be set to "Fixed". If the bug report is a duplicate of another issue, the resolution should be set to "Duplicate". If the bug cannot be reproduced, the resolution should be set to "Cannot Reproduce".

## Test Reporting Phase

### Test Report Containment

In the Test Report, it is important to include the following items:

1. List of tasks with priority and status: This should include a detailed list of all the tests that were conducted during the current iteration, along with their priority and status (pass/fail). This will give a clear picture of the progress made during the testing phase.

2. Metrics: It is essential to calculate and track metrics such as the number of test cases executed, the number of defects found, the defect density, the test coverage, and the pass rate. The trend line for the last 5 releases should also be included to provide a historical view of the performance of the application over time.

3. Recommendations: Based on the test results, it is essential to provide recommendations on how to improve the application's quality. This can include suggestions for improving test coverage, enhancing the test process, improving the development process, or identifying areas for further testing.

4. Test Summary: A brief summary of the testing phase should be included, outlining the key findings, areas of concern, and recommendations for improvement. This section should be easy to read and provide a quick overview of the testing process.

5. Defects: A list of all defects found during testing should be included, along with their severity, priority, and status. It is important to ensure that each defect has a clear description and steps to reproduce so that it can be easily replicated and addressed by the development team.

6. Test Environment: The details of the test environment should be included in the report, including the hardware and software configurations used for testing, test data used and the tools used for testing.

7. Conclusion: The report should end with a brief conclusion summarizing the key findings, areas of concern, and recommendations for improvement. This should also include any final thoughts or recommendations for future testing.

### List of the Metrics to Track on a Project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric name | Formula | Responsible role | Frequency | Way to provide |
| DC – defect containment | DC = (Number of non-rejected defects registered during by end-users on production environment)/(Total number of non-rejected defects registered) | Testing Team Lead | after each release to production | 2nd column chart |
| TCE - Test Case Effectiveness | TCE = (Number of passed test cases)/(Total test cases executed) | QA Engineer | After each sprint | Line chart |
| TCEE - Test Case Execution Efficiency | TCEE = (Total time spent on test execution)/(Total time allocated for test execution) | Testing Team Lead | After each sprint | Bar chart |
| RC - Requirements Coverage | RC = (Number of requirements tested)/(Total requirements) | Testing Team Lead | After each sprint | Pie chart |
| TP – Test Pass Rate | (Number of passed automated tests) / (Total number of automated tests) | Automation Engineer | After each test cycle | 2nd column chart |
| TRT – Test Run Time | Total time taken to execute automated tests | Automation Engineer | After each test cycle | Table |
| TB – test coverage by automation | (Number of test cases automated) / (Total number of test cases) | Automation Engineer | After each test cycle | 2nd column chart |

# Risk Management

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Summary/ Description | Probability | Impact | Risk Exposure  (Probability \* Impact) | Assignee | Treatment Strategy | Mitigation and/or Damage Control Plan |
| Insufficient resources for testing | Medium | Moderate | Medium | Delivery Manager | Mitigation | Allocate additional resources for testing. Prioritize testing tasks based on criticality. Optimize resource allocation for testing. |
| Limited availability of test environments | High | Moderate | High | Development Team Lead | Avoidance | Request additional test environments. Optimize usage of existing test environments. Coordinate with other teams for environment availability. |
| Inadequate test coverage | Medium | High | High | Testing Team Lead | Mitigation | Review and update test requirements. Create additional test cases for uncovered areas. Prioritize test execution based on criticality. |
| High dependency on third-party APIs | High | High | Highest | Test Automation Engineer | Acceptance | Monitor and communicate with third-party API providers. Develop backup plans in case of API failures. Create alternative test scenarios for API failures. |
| Scope creep leading to timeline delays | Medium | High | High | Delivery Manager | Mitigation | Establish a robust change control process. Define clear project scope and communicate it to stakeholders. Monitor and manage scope changes effectively |
| Security vulnerabilities in application | Low | High | Medium | Security Analyst | Transfer | Conduct regular security audits and assessments. Collaborate with security team for vulnerability scanning and patching. Obtain appropriate insurance coverage for potential security breaches |
| High defect density in previous releases | Medium | Moderate | Medium | Testing Team Lead | Mitigation | Review defect trends, prioritize testing based on past issues |
| Unavailability of approved UI mockups in time | High | Moderate | High | UX/UI Designer | Mitigation | Schedule reviews in advance, have backup plans for design changes |
| Vacations, illnesses of team members | Medium | Low | Low | Delivery Manager | Acceptance | Plan for contingencies, cross-train team members, manage workload accordingly |

# Test Team

## Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Project Role | Name | Location | Responsibilities |
| 1 | Delivery Manager |  | Germany | Oversee project delivery, manage resources and schedule |
| 2 | Development Team Lead |  | Germany | Lead development team, assign tasks, ensure code quality |
| 3 | Testing Team Lead | Yana Sudakova | Czech Republic | Lead QA team, develop and implement test strategies |
| 4 | Manual QA |  | Spain | Execute manual tests, identify and report defects |
| 5 | Manual QA |  | Spain | Execute manual tests, identify and report defects |
| 6 | Manual QA |  | Spain | Execute manual tests, identify and report defects |
| 7 | Automation QA |  | Germany | Develop and maintain automated test scripts |
| 8 | Automation QA |  | Germany | Develop and maintain automated test scripts |
| 9 | Security Tester |  | Germany | Identify and report security vulnerabilities |
| 10 | Product Owner |  | Germany | Define and prioritize product requirements |
| 11 | UX Designer |  | Germany | Create user interface designs and prototypes |
| 12 | Developer |  | Boston | Develop and maintain software code, collaborate with testing team on issue resolution |
| 13 | Developer |  | Berlin | Develop and maintain software code, collaborate with testing team on issue resolution |
| 14 | Developer |  | Seoul | Develop and maintain software code, collaborate with testing team on issue resolution |

## Stakeholders

| # | Project Role | Name, e-mail, location |
| --- | --- | --- |
| Internal (inside EPAM) | | |
|  | Delivery Manager |  |
|  | Testing Team Lead | Yana Sudakova, [yana\_sudakova@epam.com](mailto:yana_sudakova@epam.com), Czech Republic |
| External (from customer side) | | |
|  | Product Owner |  |
|  | Development Team Lead |  |

## Communication Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Issue | Responsible person/people | Contact point | Communication plan |
| 1 | Inform the team about critical defect | All test team members | All project team members | Via project chat |
| 2 | Update on test progress | Testing Team Lead | Delivery Manager | Weekly status report |
| 3 | Defect status and priority updates | Manual QA, Automation QA | Development Team Lead | Daily stand-up meeting |
| 4 | Test environment availability | Testing Team Lead | Development Team Lead | Ad-hoc email/communication |
| 5 | Escalation of critical issues | Testing Team Lead, Delivery Manager | Delivery Manager | Immediate email/notification |
| 6 | Change in project timeline or scope | Delivery Manager | Entire Project Team | Formal change request process |
| 7 | Update on test data availability | Testing Team Lead | Development Team Lead | Weekly status report |
| 8 | Testing milestones and deadlines | Testing Team Lead | Entire Project Team | Project timeline or schedule update |
| 9 | Test coverage and completion status | Testing Team Lead | Delivery Manager | Weekly status report |
| 10 | Customer communication and updates | Delivery Manager | Customer | Regular email/communication |

## Escalation Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Issue | Responsible person/people | Contact point | Communication plan |
| 1 | Test team is not in schedule | Testing Team Lead | Delivery Manager | Via email |
| 2 | Critical defects found in production | Testing Team Lead, Development Team Lead | Delivery Manager | Immediate notification through email, phone or chat |
| 3 | Testing milestones are at risk of delay | Testing Team Lead, Delivery Manager | Delivery Manager | Weekly status report and project meeting |
| 4 | High severity security vulnerability | Security Tester, Development Team Lead | Delivery Manager | Immediate notification through email, phone or chat |
| 5 | Unresolved testing dependencies | Testing Team Lead, Delivery Manager | Delivery Manager | Ad-hoc communication through email or chat |
| 6 | Test environment is unavailable | Testing Team Lead, Development Team Lead | Delivery Manager | Immediate notification through email or chat |
| 7 | Test data is not available | Testing Team Lead, Development Team Lead | Delivery Manager | Weekly status report and project meeting |
| 8 | Changes in project scope or requirements | Delivery Manager | Entire Project Team | Formal change request process |
| 9 | Customer escalation or complaint | Delivery Manager, Delivery Manager | Customer | Immediate notification through email or phone |
| 10 | Resource constraints or conflicts | Delivery Manager | Delivery Manager | Project meeting and resolution discussion |
| 11 | Critical issues affecting testing progress | Testing Team Lead, Delivery Manager | Delivery Manager | Immediate notification through email, phone or chat |

# Test Schedule

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Activity | Sprint day 1 | Sprint day 2 | Sprint day 3 | Sprint day 4 | Sprint day 5 | Sprint day 6 | Sprint day 7 | Sprint day 8 | Sprint day 9 | Sprint day 10 |
| Manual test planning and design |  |  |  |  |  |  |  |  |  |  |
| Manual test execution |  |  |  |  |  |  |  |  |  |  |
| Automated test planning and design |  |  |  |  |  |  |  |  |  |  |
| Automated test development |  |  |  |  |  |  |  |  |  |  |
| Automated test execution |  |  |  |  |  |  |  |  |  |  |
| Defect reporting and management |  |  |  |  |  |  |  |  |  |  |
| **Testing on stage environment** |  |  |  |  |  |  |  |  |  |  |
| **Regression on stage environment** |  |  |  |  |  |  |  |  |  |  |
| **Regression on production environment (deployment)** |  |  |  |  |  |  |  |  |  |  |
| Test results analysis and reporting, demo |  |  |  |  |  |  |  |  |  |  |

# Test Deliverables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Artifact to be provided | Target audience | Author/ Responsible Person(s) | Frequency (delivery time) | Method of delivery | Link to the template if exists |
| 1 | Test Plans | Project Team | Testing Team Lead | Once before the testing start | Via e-mail, confirmation from Delivery Manager is needed |  |
| 2 | Test Strategy | Project Team | Testing Team Lead | Once before the testing start | Via e-mail, confirmation from Delivery Manager is needed |  |
| 3 | Test Cases | Manual QA Team | Manual QA Team | Before the testing start | TestRail |  |
| 4 | Bug reports | Project Team | QA Team | Upon finding a bug | Jira |  |
| 5 | Test Result Reports | Project Team | Testing Team Lead | Weekly | Email |  |

| REVISION HISTORY | | | | | |
| --- | --- | --- | --- | --- | --- |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Date |
| 1.0 | Test Plan initial version | Yana Sudakova | 19.04.2023 |  |  |
|  |  |  |  |  |  |