QA manual and strategic thinking:

Introduction:

Alpha's platform provides its clients with a set of portals designed for companies that need quick access to capital investment. Alpha makes traditional and non-traditional data sources available to its clients to enhance the investment decision process.

Alpha understands the importance of quality assurance in the software development lifecycle. It is imperative that the platform is fully tested before it is made available to its customers.
At the same time, the company has an aggressive schedule for releasing new products. The testing cycle must be short, reliable and efficient.

SOFTWARE SPECIFICATIONS

Over the past five years, Alpha has built an investment platform for small businesses. This software application is responsible for managing and providing capital investment services. During the investment enrollment process (submission of the application), the client creates their login credentials (username and password) using the Alpha Client Portal. The company is also assigned a unique identifier. The platform verifies these three parameters as part of the user authentication process. Once the user has logged in, the application presents the current status of the Investment Request ("Under review", "Verification", "Document preparation", "Approved", "Denied") or the status of the investment ("Active", "Closed") if access to the investment and its benefits has already been granted.

If the investment is active, the application displays its profit balance, previous payment activity, pending payments, or new investment amounts. The user can initiate proactive payments to their investment. Proactive payments can be made using a credit card or a new or existing bank account. The application keeps track of previously used funding accounts (credit cards or bank accounts) and presents them as a possible option to the user. Funding accounts are authorized each time a payment is submitted.

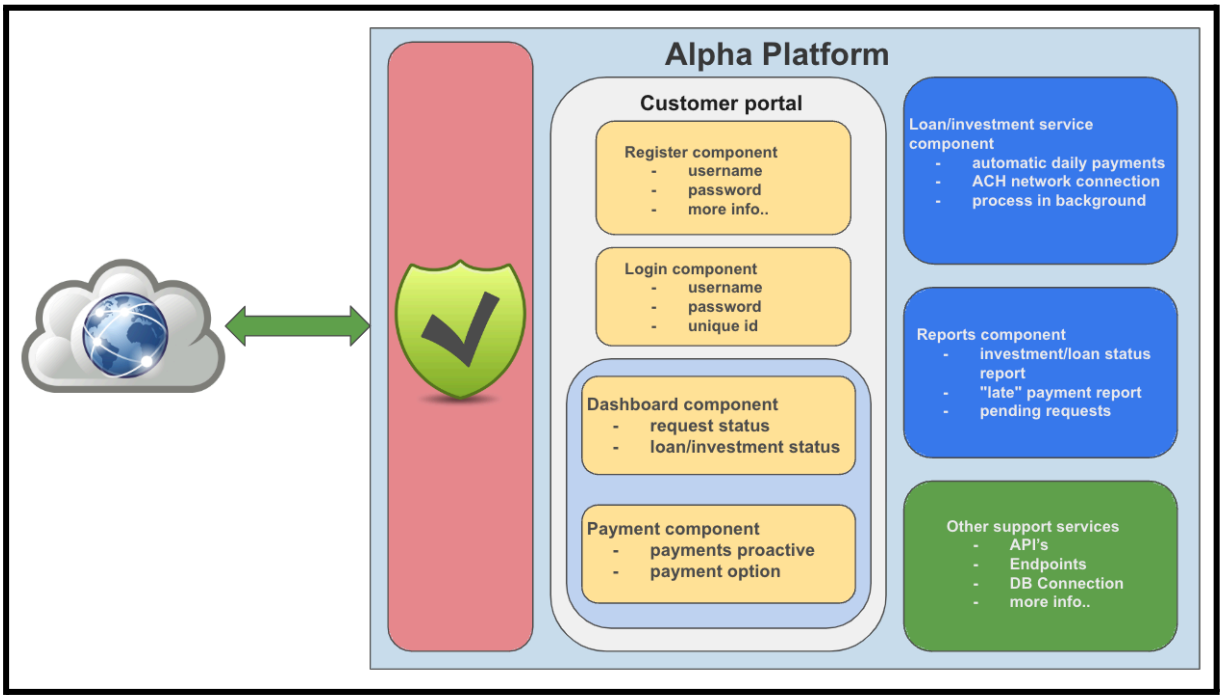
Alpha plans to sell its platform in other countries. The platform can be easily adapted to various languages and regions without any software changes. For example, the application information can be displayed in the language and currency chosen by the company. After an investment is established, Alpha's Investment service (JobAgent) begins to collect/deliver payments daily as scheduled. The fundraising process is automatic. The application creates a daily file with all outstanding or credited payments. This file is sent to the ACH network, and the funds are ultimately deposited into the Alpha client's or investor's bank account, depending on whether it is a debit or credit collection. The next day, Alpha receives a file with all transactions that could not be processed. The investment service processes this file and updates the payment/collection status accordingly. The platform provides the ability to generate multiple types of reports, including: investment status report, "late" payment report, pending requests, etc.

ITEMS TO EVALUATE
Your task is to present a quality assurance plan to the Alpha team. Please address the following elements:

1. What QA methodology would you establish?
2. What types of tests would you have your team perform?
3. Create a list of test cases or scenarios as you see fit, according to the software specifications.
4. Define the test plan for this project.
5. What tools/software would you recommend using to execute the tests?
6. How would you approach test automation on this project considering the complexity of the platform and that there is 5 years of functionality that has not been automated?
7. How would you transition the current QA team from performing functional testing to automation without sacrificing quality and speed?
8. How would you implement a continuous delivery flow in production?
9. What processes, techniques and tools would you use to perform QA automation for cross-browser and cross-device testing as effectively as possible?

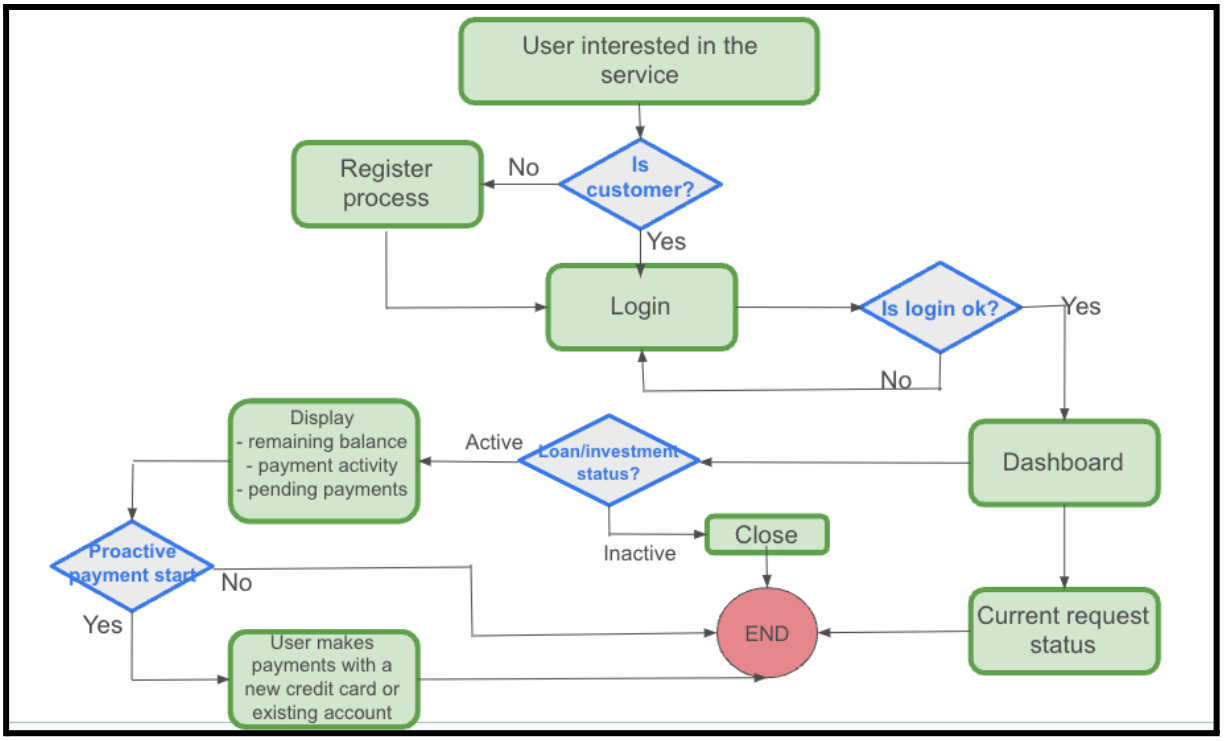
**ANALYSIS INFORMATION**

**IT Diagram**

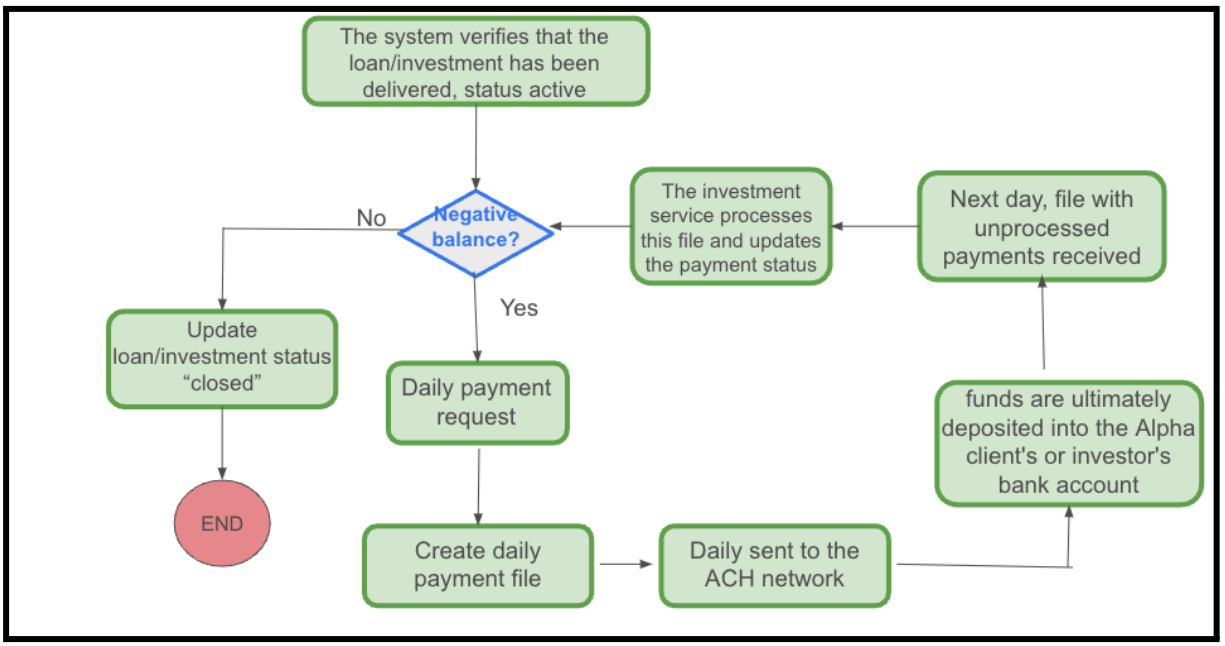


## Activity Flowchart

### The User




### The System



**DETAILED SPECIFICATIONS**

**Business Requirements**

- The enrollment process must be from the customer portal, from here you create your login credentials.
- The collection process must be automatic.
- The Loan Servicing component collects/delivers payments on a daily basis.

**Rule Requirements**

- The user can have two types of objects, a request or a loan.
- The user can initiate proactive credits/payments to his investment.

**User specifications**

Stakeholder requirements

- We need the application to create a daily file with all payments pending or paid.
- We need that file to be sent to the ACH network and the funds eventually deposited into the Alpha customer's or investor's bank account.
- We need the next day Alpha to receive a file with all transactions that could not be collected.
- We need the loan servicing component to process this file and update the loan status.

## Software requirements specifications

**Functional requirements**

**Process**

- The customer creates its login credentials (username and password) using the Alpha Customer Portal.
- The company is also assigned a unique identifier for each user.
- When the user logs into the platform, the platform displays the current status of the application (“Under Review”, “Verification”, “Document Preparation”, “Approved”, “Denied”) or the status of the loan (“Active”, “Closed”) if a loan has already been granted.
- If the loan is active, the application displays the loan's earnings balance, previous payment activity, pending payments or new amounts to be invested.
- The user can initiate proactive payments to their loan.
- Proactive payments can be made using a new or existing Credit Card or Bank Account.
- The application keeps track of previously used funding accounts (Credit Cards or Bank Accounts) and presents them as a possible option to the user.
- Funding accounts are authorized each time a payment is submitted.
- After a loan has been granted and funded, the Loan Servicing component of Alpha begins to collect/make payments on a daily basis. The collection process is automatic.

**Security**

- The system will control access and allow access only to authorized users.

- Users must log into the system platform with a username and password and a unique identifier.

**Graphical user interface**

- The login screen should have three text boxes with the following labels: User, password and unique identifier, the last two with hidden text (asterisks).
- Colors, styles, platform field limits, quantity and type of data, etc. are not detailed.

**Non-functional requirements**

- The solution must be easily adaptable to multiple languages and regions without software changes.
- The solution must be able to handle up to 5,000 concurrent users without performance degradation.
- All communications with external systems must be encrypted.
- The probability of system failure must not exceed 0.05.
- The platform provides the ability to generate multiple types of reports including: loan status report, “Late” payment report, pending applications, etc.

## Information not detailed

Important points are not detailed, such as the language of the application, the type of database used (relational, non-relational), the treatment of sensitive data, the budget and the human resources available. In addition, the acceptance criteria are not explicit and it is not specified whether the user can have a request and a loan at the same time.

## 1. What QA methodology would you establish?

Given the context provided in the text, where the Alpha platform recognizes the importance of quality control in the software development life cycle and the need for thorough testing before launching the platform to its customers, it would be advisable to establish a quality control methodology that ensures reliability, efficiency and speed in the testing process, taking into account the aggressive schedule for launching new products. I recommend using the Agile methodology (Scrum).

The Benefits of the SCRUM Methodology

- Eliminating errors or correcting them is considerably easier with the SCRUM methodology.
- Easy visibility of all stages of the process throughout its development.
- It is easy to adapt to changes required by customers, since the SCRUM methodology consists of short sprints with constant feedback.
- You can change the development at any stage by having access to increased flexibility with the SCRUM methodology.
- By involving customers, the SCRUM methodology ensures the best results.

- It enables success in businesses where documentation is difficult.
- Fast results and a simple testing procedure for better production and work quality.
- It is a lightweight method that involves frequent progress updates. Updates are achieved through regular meetings.
- It is iterative in nature and needs continuous user feedback for process improvement.
- Customers have access to a transparent process that allows them to track the entire procedure and measure individual productivity.
- The product is usually delivered in a timely manner.
- The SCRUM methodology is quite economical and delivers desired results in a very short time.

The Disadvantages of the SCRUM Methodology

- If a team member leaves the process during development, it has a negative impact on the development of the project.
- With no deadline to deliver the product, it leads to uncontrolled scope and always keeps project managers demanding new features to be delivered.
- Does not come with any expected time limits and cost assessments, which can cause it to expand to multiple sprints.
- Requires strong commitment from all team members. In case some members are not up to the task, it can cause serious problems.
- Mostly suitable for small teams that have strong cohesion and understanding.
- Requires a group of experienced people to carry out the tasks.
- To quantify the process, the test team must perform regression testing of the development after each sprint. This makes it difficult to implement the SCRUM methodology.

## 2. What types of tests would you have your team perform?

I would implement two types of testing; manual and automated, with black box or white box approaches depending on the need of what is to be tested:

**Functional**

- Unit Tests: These are the responsibility of the developers, but the QA team must be trained to perform them.
- Component Testing: each functionality is tested separately.
- Integration Testing: mainly focused on testing components working in an integrated way.
- Acceptance Testing: the QA team must accompany the product owner in these tests to accept and approve the functionality.
- Smoke Testing: these tests are performed in production after new deployments, they can be manual or automated, the basic flows of the application are performed without generating logs that affect the reports or the indicator.

**Non-functional**

- Stress: stress testing refers to a type of testing so hard that it is expected to lead to program failure.

- Load: load testing is a type of performance testing that determines the performance of a system, software product or software application under load conditions based on real life.
- Usability: In the case of this platform, this type of testing is intended to verify the usability of the system.
- Performance: is a testing practice that determines the performance of a system in terms of response and stability under a specific workload.

### 3. Create a list of test cases or scenarios as you see appropriate, according to the software specifications.

It is convenient to have a complete description of the scenarios and according to these generate the test cases, however, with the requirements we can have a version of these, the test case is the smallest unit of the test plan, it includes a description of the actions and parameters necessary to achieve and verify the expected behavior of a function in the test.

**Types of test cases**

- **Positive**: Tests intended to verify the correct operation of the functionality using the correct input format.
- **Negative**: Tests intended to verify the correct operation of the functionality using the incorrect input format, expecting an error message as a result.
- **Limit value**: Tests intended to verify the correct operation of the functionality using allowed and disallowed formats.

### List of test cases

- TC001 - Register a customer with complete documentation/data.
- TC002 - Register a customer with incomplete documentation/data.
- TC003 - Register a customer with invalid username/password/email or out of parameters.
- TC004 - Log in with a registered user.
- TC005 - Log in with an unregistered user.
- TC006 - Login with correct and incorrect registered user security data.
- TC007 - Login with correct and empty security data of a registered user.
- TC008 - Login with empty user and correct security data.
- TC009 - Logout.
- TC010 - Verify that the dashboard is displayed.
- TC011 - Verify application status for a customer with an active request.
- TC012 - Verify application status for a customer with an inactive request.
- TC013 - Verify loan status for a client with an active loan.
- TC014 - Verify loan status for a client with a closed loan.
- TC015 - Verify active loan consolidated information for a client with an active loan.
- TC016 - Verify the consolidated information of an active loan for a client with a closed loan.
- TC017 - Initiate a proactive payment with a client with an active loan.
- TC018 - Initiate a proactive payment with a customer with an inactive loan.

- TC019 - Initiate a proactive payment with a customer using a new credit card or bank account.
- TC020 - Initiate a proactive payment with a customer using an existing credit card or bank account.
- TC021 - Initiate a proactive payment with a customer using an invalid credit card or bank account.
- TC022 - Initiate a proactive payment with a customer using an empty credit card or bank account.
- TC023 - Initiate a proactive payment with a customer using an existing credit card or bank account with an empty payment amount.
- TC024 - Initiate a proactive payment with a customer using an existing credit card or bank account with zero amount payment.
- TC025 - Initiate a proactive payment with a customer using an existing credit card or bank account with negative amount payment.
- TC026 - Initiate a proactive payment with a customer using an existing credit card or bank account with loan payment amount greater than loan.
- TC027 - Verify that the funding account is in authorized status after a payment.
- TC028 - Verify that the funding account does not change status after a failed payment.
- TC029 - Verify that after loan dispersal, the loan status changes to active.
- TC030 - Verify that the Loan Servicing component of the alpha platform begins collecting payments daily after loan dispersal.
- TC031 - Verify that the Loan Servicing component of the alpha platform has stopped collecting loan payments when the loan status is "closed".
- TC032 - Verify that the collection process is automatic.
- TC033 - Verify that the automatic collection process stopped when the loan status is "Closed".
- TC034 - Verify the creation of a daily file of all overdue payments.
- TC035 - Verify daily file submission to the ACH network.
- TC036 - Verify response for sending corrupt files to the ACH network.
- TC037 - Verify response for sending empty files to the ACH network.
- TC038 - Verify response from sending files with invalid data to the ACH network.
- TC039 - Verify payment consolidation.
- TC040 - Verify that the loan administration component processes files.
- TC041 - Verify that the loan administration component processes empty files.
- TC042 - Verify that the loan administration component processes corrupted files.
- TC043 - Verify that the loan administration component processes files with invalid data.
- TC044 - Verify that the loan administration component updates the status of payments.
- TC045 - Verify that the language settings are correctly applied.

## 4. Define the test plan for this project.

**Agile testing plan**
**Project: Alpha Platform**
**May - 2024**

**Introduction**

Alpha's platform provides its clients with a suite of portals designed for companies that need quick access to capital investment. Alpha makes traditional and non-traditional data sources available to its clients to enhance the investment decision process.

Alpha understands the importance of quality assurance in the software development lifecycle. It is imperative that the platform is fully tested before it is made available to its customers.
At the same time, the company has an aggressive schedule for releasing new products. The testing cycle must be short, reliable and efficient.

**Test objectives:**

- The objective of the testing is to verify the functionality of the alpha platform, the team should focus on testing operations related to the allocation, dispersion, payment and status change of client loans, as well as site performance.
- Detects defects in the software.
- Verify appropriate integration of components.
- Verify that all requirements have been implemented correctly.

**Scope:**

**In the scope of this project:**

All Alpha platform features that were defined in the software requirements specifications need to be tested.

- Registration module for platform users
- Login module
- Dashboard
- Loan Servicing Component
- Payment Component
- Reporting component

**Out of scope:**

These features will not be tested because they are not included in the software requirements specifications:

- User and Software interfaces
- Hardware components
- Database Logic
- Web site security and performance

**Testing Types:**
In the alpha platform, the following types of tests should be carried out:

- Functional Testing: ensures that all features work from start to finish.
- Component Testing: each functionality is tested separately.
- Integration Testing: mainly focused on testing components working in an integrated way.
- Acceptance Testing: the QA team must accompany the product owner in these tests to accept and approve the functionality.
- Smoke Testing: these tests are performed in production after new deployments, they can be manual or automated, the basic flows of the application are performed without generating logs that affect the reports or the indicator.
- Stress testing: stress testing refers to a type of testing so hard that it is expected to lead to program failure.
- Load testing: load testing is a type of performance testing that determines the performance of a system, software product or software application under load conditions based on real life.
- Performance testing: is a testing practice that determines the performance of a system in terms of response and stability under a specific workload.

**Test Criteria  and acceptance metrics:**

**Suspension Criteria**

- If team members report that 25% of the test cases have failed, testing is suspended until the development team corrects all failed cases.

**Acceptance Criteria**
Specifies the criteria that denote successful completion of a test phase.

- The run rate must mandatorily be 100%, unless a clear reason is given.
- The pass rate is 80%, reaching it is mandatory.

**Testing's deliverables:**

Test deliverables are provided as follows:

Prior to testing phase:

- Test plans document.
- Test case documents.
- Test design specifications.

During testing:

- Test tools.
- Test data.

- Test traceability matrix.
- Error and execution logs.

After test cycles have been completed:

- Test results/reports.
- Defect report.
- Installation/test procedure guidelines.
- Release notes.

**Risk and Issues:**

| RISK | IMPACT | PROBABILITY | MITIGATION |
|---|---|---|---|
| Team members lack the required skills for platform testing. | Medium | Low | Plan a training course to improve the skills of your members. |
| The project schedule is too tight. | High | High | Set test priority for each test activity. |
| Test Manager has poor management skills. | High | Low | Plan leadership training for the manager. |
| Lack of cooperation negatively affects employee productivity. | Low | Low | Encourage each team member in their task and inspire them to greater efforts. |
| Incorrect budget estimation and cost overruns. | High | Low | Establish the scope before starting work, pay close attention to project planning, and constantly track and measure progress. |
| Unfriendly, unclear, or difficult-to-use interfaces. | Low | Low | Conduct surveys and act on user feedback. Maintain a user-centered approach. |

**Test Resources:**

Human Resources

- **QA Manager**: will be responsible for managing the whole project, Define the project directions, acquire the appropriate resources, build and ensure the test environment and assets are managed and maintained, support the Tester to use the test environment for test execution.

- **Tester**: Verify if test process meets specified requirements, Identify and describe appropriate test automation techniques/tools/architecture, test case design, verify and evaluate Test Approach. also, Execute tests, record results, report defects.

System Resources

- Server Cloud server
- Tools to track bugs, task boards; have a Test Tool that can automatically generate test results in the predefined format and automated test execution.
- Internet connection with speed higher than 25Mb
- Computers with competitive hardware resources and good performance.
- Smartphone with Android or iOS

**Test Environment:**

The test environment should be an environment with the same software features as the production environment, not necessarily with the same capabilities (this would increase costs considerably).

- The QA team should have access to all functionality.
- There should be no sensitive information.
- There should be no relationship between the test environment and the production environment.

**Bug Report:**
In order to have a better traceability of reported bugs, their progress and their interaction, bugs will be reported in new tickets that will follow a set of standards. This will replace the report with comments.

> ////////////////////////////////////////////////
> *# Description/summary*
> *If you feel the name is not sufficient, explain the bug in a few words. Share it in easy-to-understand language. Keep in mind that your description might be used to find this ticket easily, so make sure to use the right words.*
>
> *## Steps to reproduce*
> *Make sure to describe, with as much detail as possible, the steps you took before you encountered the bug. Include the user credentials used when the bug was identified (role and permissions might affect the outcome).*

*## Severity*
*- **Critical:** Bug capable of triggering complete system shutdown or blocking the critical business functionalities.*
*- **High:** Bug capable of collapsing large parts of the system or that is blocking the flow of a critical business functionality. Also bugs  related with different behavior than the client desired.*
*- **Minor:** Results in some unexpected or undesired behavior, but not enough to disrupt system function*
*- **Low:** Bug won't result in any noticeable breakdown of the system*
*- **Enhancement:** The issue is not an error or bug itself, but a change that could improve the system in some way (usability, performance, etc.). These types of findings should be treated separately and referred to the Product Manager / Product Owner for evaluation.*

*## Priority*
*- **Urgent:** Bug must be resolved at the earliest as it affects the system adversely and renders it unusable until it is resolved. The bug is a "blocker" for the systems use.*
*- **High**: Bug must be resolved at the earliest as it affects a core feature or the system.*
*- **Medium:** Bug can be fixed in the normal course of development and testing.*
*- **Low:** Bug can be fixed at a later date. Other, more serious bugs take priority*

*## Environment*
*Depending on your browser, operating system, zoom level and screen size, websites may behave differently from one environment to another. Make sure your developers know your technical environment.*

*## Console Logs*
*By collecting the console logs your developers will find it a lot easier to reproduce and resolve any bug.*

*## Source URL*
*Make it easy for your developers spot the problem by including the URL of the page where you found the bug. Big time saver!*

*## Evidence*
*A picture is worth a thousand words. Although it might not be enough, a visual element like a screenshot or a video will help your developers understand the problem better and faster.*

*## Expected  Results vs Actual Results*
*//////////////////////////////////////*

5. **What tools/software would you recommend using to run the tests?**

**Testing Tools:**

- Test management tools: **Jira**.
- Functional testing: **Playwright and Postman.**
- Integration Testing: **IDE supported by the programming language.**
- Smoke Testing: **Playwright.**
- Load, Performance and Stress: **Jmeter**

6. **How would you approach test automation on this project considering the complexity of the platform and that there is 5 years of functionality that has not been automated?**

I would approach this with the creation of an automation strategy.

Analysis and prioritization of the tests to be automated.
We start by understanding and identifying the critical areas, core functionalities and key workflows. prioritizing the activities by role, this prioritization is determined by the impact each functionality has on the business, then we determine which of these activities or workflows are repeatable, complex and how many priority points will be assigned.
Then, the functionality will be gradually automated according to the priority score.

**Scope**.
Automated black box testing is performed on the prioritized functionalities until the maximum possible coverage is reached, this activity is re-evaluated in each sprint.

Roles and responsibilities

1. QA Manager: will be responsible for automated test planning and tracking.

2. Automation QA Tester: will be responsible for test design and implementation, automated test execution and test results reporting.

3. Stakeholders: will be in charge of decision making.

**Riesgos.**

| RISK | PROBABILITY | IMPACT | CONTINGENCY PLAN |
|---|---|---|---|
| Changes in functionalities with high cohesion to the business may cause instability in the | Low | High | Re-plan the functions to be automated |

| | | | |
|---|---|---|---|
| platform. | | | |
| Requests for changes in those functionalities that already have automated test cases. This causes rework because these scripts need to be updated. | Medium | Medium | Estimate the timing of the change and reprioritize the list of functions to be automated in the sprint. |

**Environment and Testing Tools**

**Environment**.
The environment in which these tests will be conducted is the User Acceptance Testing (UAT) environment, provided that sensitive data is neutralized.

**Tools**.
playwright: because it is a powerful tool, easy to implement and with multi-browser and multi-device properties.

**Automation architecture**
We will use the POM design pattern with playwright, with the goal of improving the readability, maintainability and reusability of the test automation code. We will create classes that represent the pages of the application, These classes will contain the page elements and the actions associated with them.
By defining page objects that encapsulate the elements and actions of each page, code reusability is promoted. This means that if there are changes to the user interface, only the corresponding page object will need to be updated instead of modifying all related tests.

**Input and Output criteria**

**Input criteria**

- Functionalities must be deployed in the QA environment and have been manually tested.
- The test framework is installed and ready for execution.
- The QA environment is available.
- Critical defects found during manual testing have been resolved and closed.

**Output criteria**

- Execution of all automated test cases.
- Sufficient coverage of the requirements and functionalities under test has been achieved.
- There are no open high severity defects.

**Test execution planning**

- List of features to be automated per Sprint.
- It is necessary that the functionalities to be automated are developed, implemented and tested manually so that they have a certain level of stability when the automation tasks are started.

**Test report**

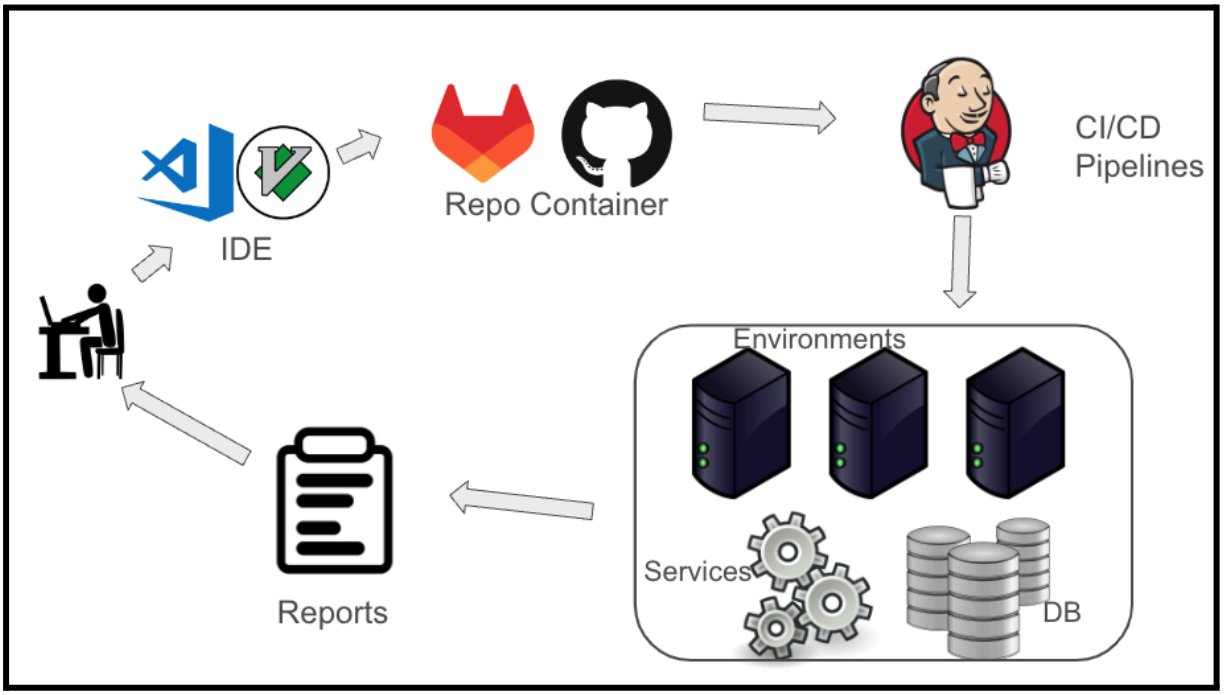
The automated test report will be obtained through Playwright. This report will report the results of the execution of each test case, it will include the tests that passed and failed, the errors encountered and the elapsed time. on each browser executed.

## 7. How would you transition the current QA team from performing functional testing to automation without sacrificing quality and speed?

I would propose training and education in test automation tools and automated test development techniques. This will help improve the team's skills and prepare them for the shift to automation, it also helps identify which team members have specific software development skills.

Choose one or more automation tools according to the technologies used and establish standards and best practices for automated test development, including code structure, dependency management, robust test writing and results reporting. This will help maintain consistency and quality throughout the automation process.

## 8. How would you implement a continuous delivery flow in production?



The data flow through the scenario is as follows:
- A change is made to the application source code.
- The application code is committed to the source code repository.
- Continuous integration triggers application build and unit testing using a comprehensive test management tool.
- Continuous deployment within Pipelines triggers automated deployment of application artifacts with environment-specific configuration values.
- Artifacts are deployed to the application service.
- Tools are used that collect and analyze health, performance and usage data.
- Developers/Tester/manager monitor and manage health, performance and usage information.
- Backlog information is used to prioritize new features and bug fixes using an available dashboard.

9. **What processes, techniques and tools would you use to perform QA automation for cross-browser and cross-device testing as effectively as possible?**

For the alpha platform, I recommend Playwright for these tests because it natively provides great compatibility with multiple modern browsers and devices, libraries and operating systems, and supports multiple programming languages (TypeScript, JavaScript, Python, .NET and Java). Automation teams can quickly adapt to the tool as it is very easy and intuitive.