Rise automation test project

**To put in a client name select all (ctrl A) then f9Code Star Academy – Kaopiz Software**

**Automation Test Plan**

**Created by: Hoan Tran**

**Version: 1.0**

**December, 01st , 2022**

**Table of Contents**

[1. INTRODUCTION 3](#_Toc154080965)

[2. TEST OBJECTIVES 3](#_Toc154080966)

[3. TEST STRATEGY 3](#_Toc154080967)

[*3.1.* *Scope of Testing* 3](#_Toc154080968)

[3.1.1. In scope testing 3](#_Toc154080969)

[3.1.2. Out of scope testing 3](#_Toc154080970)

[*3.2.* *Test Levels and Test Types* 4](#_Toc154080971)

[*3.2.1.* *Test levels* 4](#_Toc154080972)

[*3.2.2.* *Test types* 4](#_Toc154080973)

[*3.3.* *Risks* 5](#_Toc154080974)

[4. TEST CRITERIA 5](#_Toc154080975)

[*4.1.* *Entry Criteria* 5](#_Toc154080976)

[*4.2.* *Suspension Criteria* 6](#_Toc154080977)

[*4.3.* *Exit Criteria* 6](#_Toc154080978)

[5. RESOURCE PLANNING 6](#_Toc154080979)

[*5.1.* *System resource* 6](#_Toc154080980)

[*5.2.* *Test Resource* 7](#_Toc154080981)

[6. SCHEDULE AND ESTIMATION 7](#_Toc154080982)

[7. TEST DELIVERABLE 7](#_Toc154080983)

[8. AUTOMATION TESTING TOOL AND STRATEGY 9](#_Toc154080984)

[*8.1.* *Testing process* 9](#_Toc154080985)

[*8.2.* *Automation strategy* 9](#_Toc154080986)

[*8.3.* *Automation testing framework* 9](#_Toc154080987)

[9. LIFECYCLE OF BUG AND RISK 10](#_Toc154080988)

1. **INTRODUCTION**

The purpose of the project is to build the system that *[project description]*.

The Test Plan determine the scopes, objectives and risk of testing. Defining the overall test approach, the overall framework, enviroment that will support test activities, resolution for risk and contingency, test criteria, timeline and resource planning.

Build automation test scripts which using for regression test phase.

1. **TEST OBJECTIVES**

* Ensure that system functionality works as expected without any critical, high and medium bugs
* Ensure that the response time when using system meets expectation
* Ensure that the integration between modules and systems work as expectation

1. **TEST STRATEGY**
   1. *Scope of Testing* 
      1. *In scope testing*

In scope items will be system testing as usability of bellow features.

The following is the list features to be tested in this project:

|  |  |  |
| --- | --- | --- |
| No. | UserStory/Function | Degree of complex |
|  | Add new task | 2 |
|  | Edit task | 2 |
|  | Delete task | 2 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Degree: 1 – Complex, 2 – Medium, 3 – Low

* + 1. *Out of scope testing*

There are some out of scope items:

* Database test
* Mobile test
* Performance test
* Security test
* API test
  1. *Test Levels and Test Types*
     1. *Test levels*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test level** | **Method** | **People in charge** | **Note** |
| 1 | Integration test | Manual/Automation | Tester | * Testing the integration of systems and packages; testing interfaces to external organizations * Test to find defects in the interfaces and in the interactions between integrated components or systems |
| 2 | System test | Manual/Automation | Tester | Verify that the whole system meets with requirements |
| 3 | User acceptance test | Manual/Automation | Client | Client test the system to make sure that the system is met the requirement. To make sure the useablitiy, maintanceability, reliablity. |

* + 1. *Test types*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test type** | **Method** | **People in charge** | **Notes** |
| 1 | Review requirement (static testing) | Manual | Tester | To verify the correction and level of detail of User Story |
| 2 | Review code | Automated/Manual | Dev team | Self-review, Peer Review |
| 3 | Functional testing (dynamic testing) | Manual | Tester/Dev team | Verify that the system meets functional requirement |
| 4 | Non-functional testing | Manual | Tester | Performance test, API test, Security test |
| 5 | Re-testing | Manual | Tester/Dev team | Testing after fixing bug to confirm the bug is removed |

* 1. *Risks*

|  |  |  |
| --- | --- | --- |
| **No.** | **Risk** | **Mitigation** |
| 1 | **Technical Risks:**  -Technology Stack Compatibility: The chosen technology stack may not be compatible with certain features or may pose integration challenges.  -Performance Issues: The application may experience performance issues, such as slow loading times or unresponsive user interfaces, especially when filtering large datasets. | * Engage with technical experts or consultants to evaluate the suitability of different technology options based on the project's specific needs. * Perform prototyping or proof-of-concept exercises to validate the compatibility of the chosen technology stack with the project requirements before full-scale implementation. * Regularly monitor updates and advancements in technology to identify opportunities for optimizing compatibility and integration with the chosen stack. * Monitor system performance in real-time using performance monitoring tools and establish alerts for abnormal behavior or performance degradation. * Continuously optimize and fine-tune the application performance based on user feedback and usage patterns. |
| 2 | **Security Risks:**  Data Breach: There's a risk of unauthorized access to sensitive user data stored within the application, especially if security measures are not adequately implemented. | * Implement multi-factor authentication (MFA) to add an extra layer of security to user accounts, requiring users to provide multiple forms of verification to access their accounts. * Utilize strong password policies and regularly prompt users to update their passwords to reduce the risk of unauthorized access due to weak or compromised credentials. * Implement access controls and role-based permissions to restrict access to sensitive data only to authorized users with a legitimate need-to-know. * Regularly review and update security measures to address emerging threats and vulnerabilities, including conducting security audits and penetration testing. |
| 3 | **Operational Risks:**  Server Downtime: The application may experience server downtime due to technical issues or maintenance, leading to interruptions in service availability. | * Utilize load balancers to distribute incoming traffic across multiple servers or data centers, ensuring that if one server fails, others can handle the load seamlessly. * Set up alerts and notifications to promptly notify system administrators or DevOps teams of any anomalies, performance degradation, or impending downtime. * Conduct maintenance during off-peak hours or scheduled maintenance windows to minimize the impact on users and reduce the risk of unexpected downtime. * Establish redundant backup systems, data replication, and off-site backups to ensure data integrity and availability in case of server failures or data loss. |
| 4 | **User Acceptance Risks:**  -User Adoption: Users may not fully adopt or engage with the application if the user interface is not intuitive or if the filter option does not meet their needs.  -Feature Overload: The filter option may include too many criteria or be overly complex, leading to confusion and dissatisfaction among users. | * Prioritize user experience (UX) design by conducting user research, usability testing, and gathering feedback throughout the development process. * Incorporate user feedback and iterate on the design to address usability issues and improve the overall user experience. * Offer interactive tutorials, walkthroughs, or video guides to demonstrate how to use the filter option effectively and maximize its benefits. * Regularly analyze usage metrics, user feedback, and support tickets to identify pain points, usability issues, and areas for enhancement. * Iterate on the filter option based on user feedback and data-driven insights to ensure it meets users' evolving needs and expectations over time. |
| 5 | **Schedule Risks:**  -Delays in Development: Unexpected technical challenges or scope changes may lead to delays in the development timeline, impacting the overall project schedule.  -Resource Constraints: Limited availability of resources, such as developers or testing environments, may result in delays or bottlenecks in the project workflow. | * Develop contingency plans and mitigation strategies for addressing identified risks, including alternative approaches, resource allocation adjustments, and buffer timelines. * Break down the project into smaller, manageable tasks or user stories with clear priorities, allowing for incremental development and regular delivery of value. * Conduct regular sprint planning sessions, retrospectives, and daily stand-ups to monitor progress, identify impediments, and make timely adjustments as needed. * Encourage regular collaboration and coordination between cross-functional teams to address technical challenges, resolve issues, and align on project priorities * Consider outsourcing or hiring additional resources, such as contractors or freelancers, to augment internal teams and address resource constraints during peak periods or critical project phases. * Encourage knowledge sharing and cross-training initiatives to build resilience and flexibility within the team, enabling members to contribute effectively across different project areas and roles. |

1. **TEST CRITERIA**
   1. *Entry Criteria*

The entry criteria refer to the conditions in order to start testing activities (test design, test execution):

1. The specification of a feature is approved and available on Jira before starting test design
2. Development completed (status must be Ready for test)
3. Features under test are already on QC environment
4. Test cases, test data are peer-review before test execution
   1. *Suspension Criteria*

The suspension criteria refer to the conditions that test activities will be suspended if suspension criteria are met during testing:

1. Any main flows are not covered
2. Main feature does not meet acceptance criteria
   1. *Exit Criteria*

The exit criteria are the targeted results of the test that need to be met to complete a test phase:

1. 100% of Test cases executed
2. No Critical, high, medium bug remaining
3. 5 % leakage (total bugs found by client / total bugs) : lọt lỗi sang khách hàng: khách hàng tìm được 5 bugs / 100 bugs team dự án tìm
4. **RESOURCE PLANNING**
   1. *System resource*

|  |  |  |
| --- | --- | --- |
| No. | Resources | Description |
| 1 | Server | * Servers: *[list out if have]* |
| 2 | Test tool | * Snagit: Provide evident for bugs * Jira, Zephyr: Task, bug tracking, Quality monitoring, Test case management, Test execution |
| 3 | Computer | * The PCs with hardware, software required |
| 4 | **Mobile device** | * *[list out iOS or android version if have]* |
| 5 | **Web browser** | * *[list out Browser version if have]* |

* 1. *Test Resource*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Start date** | **End date** |
| Chau Tran | Tester work 50% | 25-Nov, 2021 | 10-Jan, 2022 |
|  | Tester work 100% | 25-Nov, 2021 | 10-Jan, 2022 |
|  | Tester work 100% | 25-Nov, 2021 | 10-Jan, 2022 |

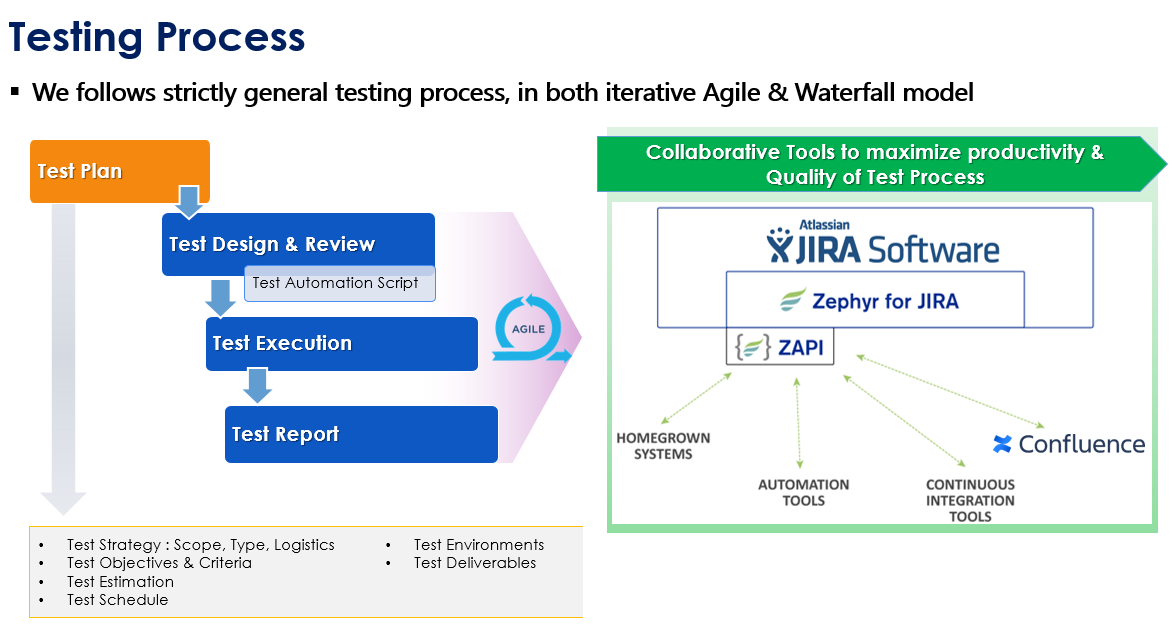
1. **SCHEDULE AND ESTIMATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Task** | **Member** | **Estimation Effort**  **(Man-day)** |
| 1 | Create test plan | Team | 1 |
| 2 | Study & review requirements |  |  |
| 3 | Create test design, test cases |  |  |
| 4 | Test execution |  |  |
| 5 | Test report & Quality Control |  |  |

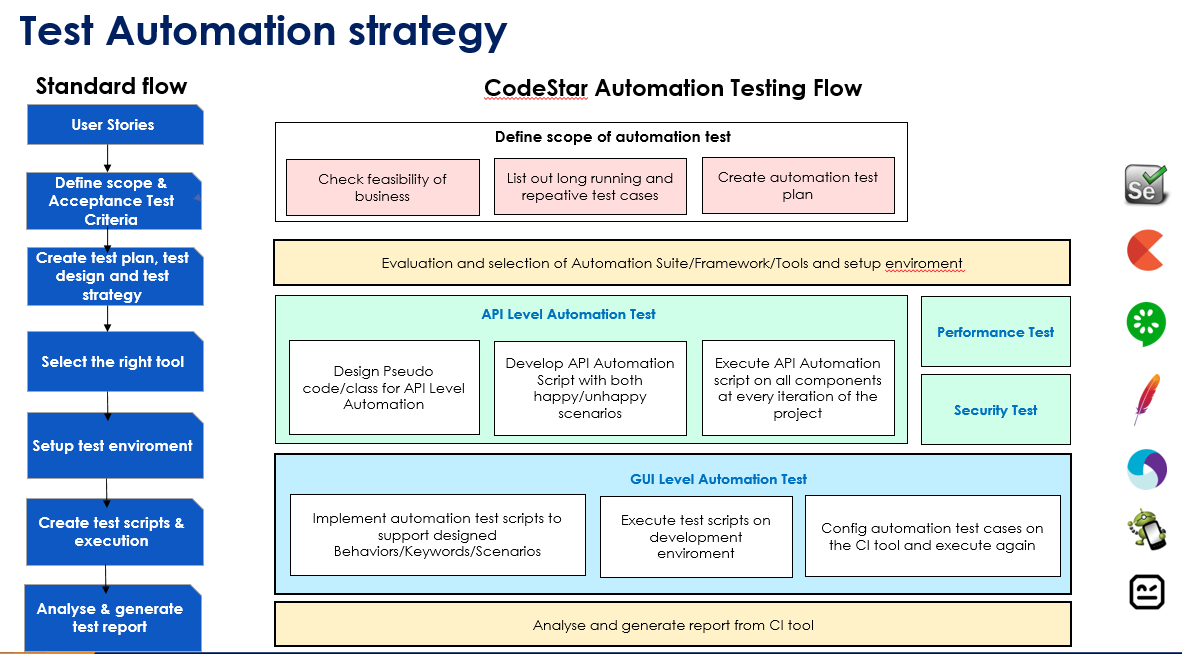
1. **TEST DELIVERABLE**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Deliverable Name** | **How to access** |
| **Before testing** | Test Plan | Test plan will be uploaded on Confluence  <https://conf.codestar.com/display/CDIS/Test+plan> |
| Test cases documents | All test cases will be uploaded on Confluence and Zephyr which is an Jira’s Add-On.  Link Confluence:  <https://conf.codestar.com/display/CDIS/Test+cases>  - Jira filter:  Open Advanced Search, then enter below JQL:  *project = CDIS AND issue type = Test* |
| **During testing** | Test cycles | Each Sprint will have at least 1 test cycle for executing sets of test cases.  Please follow this example to filter the test cycle: |
| **After testing** | Test reports | Test reports will be created after each sprint and published on Confluence: [https://conf. codestar.com/display/CDIS/Test+reports](https://conf.gemvietnam.com/display/CDIS/Test+reports) |
| Bugs | All bugs logged on Jira   * Sprint bugs are sub-tasks of the corresponding User story * Bugs/Improvement are linked to User stories, not add into current sprint |

1. **AUTOMATION TESTING TOOL AND STRATEGY**
   1. *Testing process*



* 1. *Automation strategy*



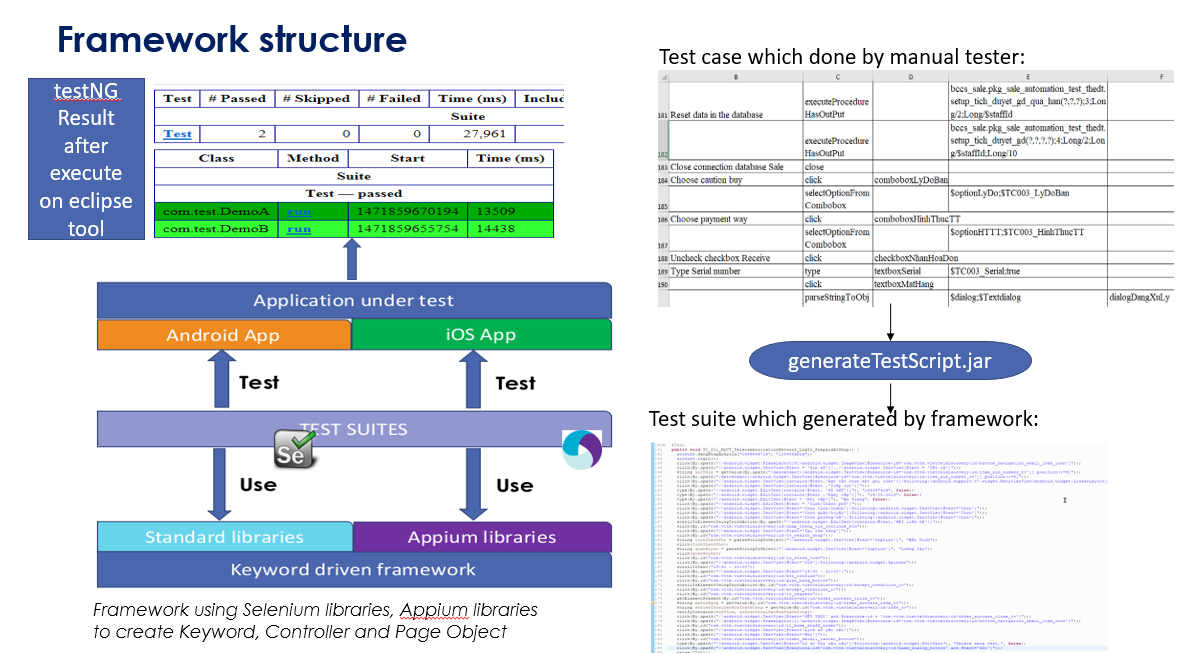
* 1. *Automation testing framework*

**Framework library/ tools:**

* Maven
* TestNG
* Github
* Selenium

**Language**:

* Java
* Python
* C#
* Javascript



1. **LIFECYCLE OF BUG AND RISK**

BUG LIFECYCLE

