**Name :** Mustafa Çevik

**Start Date :** 25.02.2022

**Test Plan**

* 1. **Objective**
     + Objectives are specified by the given task:
       - Verify that the functionality of <https://app.deel.training/login> signup screen works properly according to the specifications.
       - Automate “creating a fixed rate contract flow” with the given tools.
  2. **Assumptions**
     + First name, Date of birth and country fields are required fields.
     + For the Date of birth field, the age should be between 15-100 (both inclusive) as per requirements.
  3. **Scope (as per the given task)**
     + Prepare test cases for 3 fields for the signup flow
     + Automate the flow to create a fixed rate contract.
  4. **Environment**
     + The webapplication is live and that will be tested according to the scope.
  5. **Tools**
     + The tools chosen according to the given task are listed below on page **8** of this plan.
  6. **Strategy**
     1. Features to be tested: Signup flow and creating a fixed rate contract
     2. Manual Test: Signup Flow
        + - First name field – error guessing and state transition as Blackbox techniques
          - Country dropdown - error guessing and state transition as Blackbox techniques
          - Date of birth field – error guessing, state transition and Boundary Value Analysis as Blackbox techniques
     3. Automation: Creating a fixed rate contract flow
  7. **Entrance criteria**
     + Complete code is available. == > Already met since the app is live for this task.
     + Requirements are defined and approved. == > Assumed as approved.
     + Test cases are developed and ready. == > The required test cases are ready.
     + Test environment has been set-up and all other necessary resources such as tools and devices are available. == > Prepared according to the tools given in the task
  8. **Defect management**
     + Defect management will be executed via Jira.
     + The following bug status will be used:
       1. New (by tester)
       2. Assigned (by tester)
       3. Open (Developer)
          - Duplicate
          - Rejected
          - Deferred
       4. Fixed (Developer)
       5. Retest (Tester)
          - Verified
          - Reopen
       6. Closed
  9. **Risk Management**

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Prob. | Impact | Mitigation Plan |
| SCHEDULE  Testing schedule is tight. | High | High | * Request to change deadline and this request is approved. The deadline becomes 02.03.2022. |
| RESOURCES  Setting up the required environment and learning new required tools can be difficult | Medium | High | * Prioritize a learning strategy * Use the official documents of the tools * Request extra time (requested and approved) |
| DEFECTS  Defects that can stop testing further | Low | High | Since the tested application is already live, this has a low probability. In case occurs, contact with the task manager. |
| SCOPE  Scope completely defined | Medium | Medium | Scope is well defined |
| Natural disasters/stoppers | Low | Medium | Work in a remote environment with some alternative places to be able to move to. |

* 1. **Exit Criteria**
     + 100% Test Scripts executed
     + 95% pass rate of Test Scripts
     + No open Critical and High severity defects
     + 95% of Medium severity defects have been closed
     + All remaining defects are either cancelled or documented as Change Requests for a future release
     + All expected and actual results are captured and documented with the test script
     + All defects logged

**The 3 fields to test:**

* **First name**
* **Date of birth**
* **Country**

**Assumptions**:

* I assume that all 3 fields are required fields.
* I assume that the requirement allows an age between 15 to 100 (both inclusive) as date of birth.

| **Test Cases** | **Scenarios** | **Description** | **Steps To Execute** | **Test Data** | **Expected Results** | **Actual result** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-001  (Positive) | Validate First name field | Check validation with a valid first name with one word | 1.Enter a valid name in the “First name” field  2.Click Complete Profile button | Ahmet | There should not appear any error (eg: no red frame) |  |
| TC-002  (Positive) | Validate First name field | Check validation with a valid name which has 2 words | 1.Enter valid name with 2 words in “First name” field  2.Click Complete Profile button | Ahmet Ali | There should not appear any error (eg: no red frame) |  |
| TC-003  (Positive) | Validate First name field | Check validation with a valid name which has 3 words | 1.Enter valid name with 3 words in “First name” field  2.Click Complete Profile button | Ahmet Ali Kaya | There should not appear any error (eg: no red frame) |  |
| TC-004  (Positive) | Validate First name field | Check validation with a valid name which has 3 words with all lowercases | 1. Enter valid name which has three words in “First name” field with all lowercases  2. Click Complete Profile button | ahmet ali kaya | There should not appear any error (eg: no red frame) |  |
| TC-005  (Positive) | Validate First name field | Check validation with a valid name with a different alphabet | 1. Enter a valid “First name” which is from a different alphabet than Latin.  2.Click Complete Profile button | Иван | There should not appear any error (eg: no red frame) |  |
| TC-006  (Negative) | Validate First name field | Check the validation by leaving the field empty | 1.Keep the “First name” field empty  2. Click Complete Profile button | NA | The empty first name should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-007  (Negative) | Validate First name field | Check validation with a name having just special characters | 1.Enter a name with special characters  2.Click Complete Profile button | !-%&^$ | The first name should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-008  (Negative) | Validate First name field | Check validation with a name having some special characters with also letters | 1.Enter a name with some letters and special characters  2.Click Complete Profile button | Ahmet%$% | The first name should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
|  |  |  |  |  |  |  |
| TC-009  (Positive) | Validate Date of birth field | Check validation with a valid date | 1.Enter a valid date  2.Click Complete Profile button | 01.01.1980 | There should not appear any error (eg: no red frame) |  |
| TC-010  (Positive) | Validate Date of birth field | Check validation with a valid day in February for a leap year. | 1.Enter a valid day in February for a “leap year” in the date field.  2.Click Complete Profile button | 29.02.2000 | There should not appear any error (eg: no red frame) |  |
| TC-011  (Negative) | Validate Date of birth field | Check validation with an invalid day in February for a “not leap year”. | 1.Enter an invalid day in February for a “not leap year” in the date field .  2.Click Complete Profile button | 29.02.2001 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-012  (Negative) | Validate Date of birth field | Check validation with an invalid day in February for “a leap year”. | 1.Enter an invalid day in February for “a leap year” in the date field .  2.Click Complete Profile button | 30.02.2000 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-013  (Positive) | Validate Date of birth field | Check validation with a valid date with an age of 100. | 1.Enter a valid date with age of 100.  2.Click Complete Profile button | 10.10.1922 | There should not appear any error (eg: no red frame) |  |
| TC-014  (Positive) | Validate Date of birth field | Check validation with a valid date with an age of 15. | 1.Enter a valid date with age of 15.  2.Click Complete Profile button | 01.01.2007 | There should not appear any error (eg: no red frame) |  |
| TC-015  (Negative) | Validate Date of birth field | Check validation of a date with an age more than 100. | 1.Enter a valid date with age more than 100.  2.Click Complete Profile button | 01.01.1921 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-016  (Negative) | Validate Date of birth field | Check validation of a date with an age less than 15. | 1.Enter a valid date with age less than 15.  2.Click Complete Profile button | 01.01.2008 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-017  (Negative) | Validate Date of birth field | Check validation with an invalid month. | 1.Enter an invalid month in the date field.  2.Click Complete Profile button | 01.13.2000 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-018  (Negative) | Validate Date of birth field | Check validation with an invalid day. | 1.Enter an invalid day in the date field.  2.Click Complete Profile button | 32.01.2000 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-019  (Negative) | Validate Date of birth field | Check validation with empty date. | 1. Leave the date field empty.  2.Click Complete Profile button | NA | The empty date of birth field should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-020  (Negative) | Validate Date of birth field | Check validation with empty day. | 1. Leave the day of the date field empty.  2.Click Complete Profile button | … .01.1980 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-021  (Negative) | Validate Date of birth field | Check validation with empty month. | 1. Leave the month of the date field empty.  2.Click Complete Profile button | 01. … .1980 | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-022  (Negative) | Validate Date of birth field | Check validation with empty year. | 1. Leave the year of the date field empty.  2.Click Complete Profile button | 01.01. … | The date should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
|  |  |  |  |  |  |  |
| TC-023  (Positive) | Check and validate the Country dropdown | Check validation with a valid country from dropdown. | 1.Select a country from dropdown.  2. Click Complete Profile button | Select “Turkey” | There should not appear any error (eg: no red frame) |  |
| TC-024  (Positive) | Check and validate the Country dropdown | Check country dropdown list after typing first 2 letters. | 1. Type the fist 2 letters of the country  2. Check the countries from dropdown that all of them include the typed letters.  3. Select the country from options  4. Click Complete Profile button | Tu | * The countries in the dropdown should be the ones which have the typed letters (with “tu” in this case). * There should not appear any error (eg: no red frame) |  |
| TC-025  (Positive) | Check and validate the Country dropdown | Check country dropdown list after typing first 3 letters. | 1. Type the fist 3 letters of the country  2. Check the countries from dropdown that all of them include the typed letters.  3. Select the country from options  4. Click Complete Profile button | Tur | * The countries in the dropdown should be the ones which have the typed letters (with “tur” in this case). * There should not appear any error (eg: no red frame) |  |
| TC-026  (Positive) | Check and validate the Country dropdown | Check the selected Country after selecting different countries from dropdown  field | 1. Select a country from dropdown.  2. Select another country from dropdown  3. Select another country from dropdown  4. Click Complete Profile button | Select different countries:  - Turkey  - Spain  - Denmark | * After selecting a new country from dropdown, the field should be populated with the new selection. * There should not appear any error for the country field (eg: no red frame) |  |
| TC-027  (Negative) | Check and validate the Country dropdown | Check the validation for empty country field | 1. Leave the country field empty.  2. Click Complete Profile button | NA | The empty country field should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-028  (Negative) | Check and validate the Country dropdown | Check the validation for an entry which is not a country | 1. Make an entry other than a country  2. Click Complete Profile button | Istanbul | A different entry other than a country should not be accepted and there should be a notification error (eg: a red frame or a message) |  |
| TC-029  (Negative) | Check and validate the Country dropdown | Check the validation for an entry with some special characters | 1. Make an entry with some special characters  2. Click Complete Profile button | +%&/ | A different entry other than a country should not be accepted and there should be a notification error (eg: a red frame or a message) |  |

**Automation**

**Tools**

- The programming language : **Javascript**

- The automation tool : **WebdriverIO**

- Testing framework : **Mocha**

- Test report tool : **Allure**

- IDE **: VS Code**

- Version control tool : **git/github**

-Bug tracking tool **: Jira**

- CI/CD tool  **: Jenkins**

**Processes and Stages**

It is required to specify which tests to automate and which tests to manually test. I have mentioned this above in summary in the test plan.

We should consider the followings to decide to automate:

* The feature/app should be stable. Automating a feature which is being changed will be useless.
* The test should be repetitive. Test cases which will be executed just several times should not be automated in order to use resources more effectively.
* We should select critical functionalities to automate such as login, register, checkout or any core functionalities of the product.
* If above are done satisfactorily, we can proceed to automate more complex flows rather than simple flows.

In the given task, my responsibility is to automate “creating a fixed rate contract” flow.

I have executed the process in terms of STLC process:

* The requirements should be analyzed thoroughly. There is no requirement document here. Instead I have the task and the live application. So I understood the task and I got familiar with the application. I manually test the flow and undertand the process.
* I prepared a test plan in order to achieve the objectives.
* In order to design the test, I specified the tools and developed the necessary test cases. Since I am not familiar with some of the tools such as WebdriverIO, mocha, I made a learning plan and got familiar with them both theoretically and practically.
* Test environment is as per given task.
* I have developped the testing framework and executed the given task with automation.

In my framework I used Page Object Model in order to increase efficiency, readability and maintainability in the framework. There are 3 packages in my framework:

* **Pages:** I have my locators and functions in the classes of this package. I export the classes and use them in my test classes under specs package. This increases flexibility of the code. For instance, when a locator changes, I simply change the locator in this package. I also have functions and reusable functions in my page classes. With these functions, I can easily write my test cases under the test specs package.
* **Specs:** I add my test classes under this package. In the test classes I use “describe()” block and inside that block “it()” blocks.
  + I generally use describe() block to describe the scenario.
  + And inside the describe() block, it() blocks are individual test cases.
* **Test\_data:** I use this package to store some test data. In order to make my code more dynamic, I get some data from this package. When the data changes, I just change the data inside this package. Besides making codes dynamic, this provides me to hide the data also.

**Reports**

I used allure reports as per the task. These reports provide comprehensive visual graphics and clear and concise test results. If there is a failure, I can also have screenshots via allure. I also use spec reports. They help me about analyzing the failures I have encountered while developing the testing framework. In some cases the screenshots may not be enough to clearly point the bug. For such cases, I can also take a screencast/video so that the bug can be easily understood by developers, customers and relevant people.

**Test Types**

1. **Sanity (smoke) test:** In this test, we ascertain that the most crucial functions of the applications/software work properly. We test the core functionalities of the software. We should execute smoke test so often that we are always sure that our software has no big issues. As a best practice, we automate our smoke tests and we run the smoke suite upon each commit (some companies prefer executing a few times daily) via a CI/CD tool. Registration, login or a checkout feature for a shopping website are some features that should be in the smoke test. We should add critical functionalities into smoke test and we should also add only required tests to make it more useful.
2. **Regression test**: We execute regression test to verify that a change in a code or a change in a feature does not impact existing functionality of the application/product. As the development process is continuously execcuted, we can execute the regression suit on a daily basis. We re-run the previously executed regression tests and compare the results with the previous ones. While preparing a regression test suite, we should focus on high-priority, complex, vulnerable areas.
3. **End-to-end test:** We prepare end-to-end tests in order to ascertain that the application/software works properly through a work flow from beginning to the end. For example, a user registers to a shopping application, then the user signs in with the newly created credentials. Then the user adds some items to the cart and proceeds to checkout. Then the user successfully places order and receives the confirmation notification/message for the purchase. This flow can be an example for a typical end-to-end flow. It is better to test also the back-end functionalities during this end-to-end testings. For example, when we register to the application via UI (frontend), it is better to ascert that registration data with backend (with API and/or DB testing). In the automated task in this project is a simple end-do-end testing. I registered to the application manually (this can be also automated). Then with the registration data, I signed in the app and then created a fixed rate contract. A regular user can execute this work flow. And we test this end-to-end work flow. In order to manage effective end-to-end tests, we should prepare compact, atomic test cases that can be used in other tests.