**Flow BP**

Software Engineering Project

Application Testing Document

Eilon Benami

Eyal Almog

Hadas Atiya

Gilad Abudi

**Contents**

Chapter 1: Functional requirements testing…………………………………………………………………………..3

Chapter 2: Test integration & Deployment ……………………………………………………………..……………..7 2.1: Execution ………………………………………….……………………………………………………………..……………..7 2.2: Debugging ………………………………………………………………………………………………………..…………….10

**Functional Requirements Testing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected | Pass/Fail |
| Requirement 1.1 Good | Create a new blank worksheet | None | The app opens correctly without exceptions | Pass |
| Req. 1.1.1 Good | Save an existing working sheet | Clicking the save button, file name | File including the diagram's xml code lies on the user's device in the wanted location | Pass |
| Req. 1.1.2 Good | Open an existing working sheet | Clicking the open button, XML file path | The diagram described on the xml file opened will be loaded to the working sheet | Pass |
| Req. 1.1.2 Bad | Open an existing working sheet | Clicking the open button, **Non XML** file path | The system should show an error message as the file is not of type xml | Pass |
| Req. 1.1.2 Bad | Open an existing working sheet | Clicking the open button, file path that doesn't exist | The system should show an error message as the file doesn't exist | Pass |
| Req. 1.2 Good | Create General, Bsync, Start and Console blocks | Dragging/clicking the wanted block from the left side tool bar | The block should be visible on the working sheet | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected | Pass/Fail |
| Req 1.2.1.1 Good 1 | Define 1 payload on a start node | Clicking on a start node, and writing a payload object onto the Payload1 text box, click apply | Payload is visible in the text area designated to it | Pass |
| Req 1.2.1.1 Good 2 | Define 2 payloads on a start node | Clicking on a start node, writing 2 in the "Number of payloads" section. Press apply. Enter text in both "Payload 1" and "Payload 2". Press Apply. | Payloads are visible in the text area designated to it | Pass |
| Req 1.2.1.1 Bad | Define 0 Payloads on a start node | Writing 0 in "number of payloads" section. Pressing apply. | The number of Payloads will not be changed | Pass |
| Req 1.2.1.2 Good 1 | giving a general block a title | After clicking a general block, writing a title in the input box designated | The title will be visible on the block | Pass |
| Req 1.2.1.2 Good 2 | Changing a general block's title | After clicking a general block, writing a new title/ deleting the old title in the designated input box | The new title will be changed accordingly on the block | Pass |
| Req 1.2.1.3 Good | Writing code in general block's code area | After clicking a general block, clicking on the "Open code editor", writing some js code, pressing apply | The code will be saved and shown on the block visually | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected | Pass/Fail |
| Req 1.2.1.3.1 Bad | Writing code with wrong syntax | After clicking a general block, clicking on the "Open code editor", writing some bad syntax js code, pressing apply | A message will be shown indicating there is a syntax error. | Pass |
| Req 1.2.1.4 Good | Writing requested, waited for and blocked events on a bsync node | Clicking a Bsync node, writing in input slots on the right hand side menu | The requested ,waited for and blocked events will be saved and shown on the block | Pass |
| Req 1.4 Good | Move block from one place to another | Dragging an existing block and changing its location | The block appears on the new location | Pass |
| Req 1.5 Good | Deleting a block/edge | Pressing delete button on the editor/on the keyboard after clicking a block/edge | The block/edge will not be visible on the working sheet | Pass |
| Req 1.9 Good 1 | Executing a bp flow program with a start node not connected to anything | Creating a start node without creating edges on it, and pressing the execute button | An appropriate message will be shown, and the relevant start nodes will be colored red | Pass |
| Req 1.9 Good 2 | Executing a bp flow program with an edge that has no source/target | Pressing the execute button while there are edges that have no source\target | An appropriate message will be shown, and the relevant edges will be colored red | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected | Pass/Fail |
| Req 2.4 Good | Clicking debug after there is a bp flow program on the working sheet | Locating a bp flow program on the working sheet, clicking debug button | The editor will change to debug mode, and enable step forward, step back and stop buttons | Pass |
| Req 2.4 Bad | Clicking debug button after there is a bad bp flow program on the working sheet | Locating a bad bp flow program (edges not connected \ start node without targets\ bad code in code slots) | An appropriate message will be shown to the user, and the editor ui will not be changed to debug mode | Pass |
| Req 3.1 Good | Clicking execution button after there is a bp flow program on the working sheet | Locating a bp flow program on the working sheet, clicking execution button | The editor will execute the bp flow program according to bp semantics, and show the events selected on the output console. | Pass |
| Req 3.2 Bad | Clicking execution button after there is a bp flow program on the working sheet | Locating a bad bp flow program (edges not connected \ start node without targets\ bad code in code slots) and clicking execution | An appropriate message will be shown to the user, and there will be no execution\the execution will stop. | Pass |

**Test integration & deployment:**

1. Execution:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected Result | Actual Result | Pass/Fail |
| RequestsList: Checks that only one of the requested events that appear in one bsync block have occurred. in addition, in addition, checks that the probability to choose one of the requests in 100 runs is under 80% (from two requests). | Legal of selected request event from bsync node | strings that represent XML code of a diagram of BP Flow syntax | One event:  "Hi" or "Goodbye" | "Goodbye" | Pass |
| HelloWorld: Checks the order of event requests that occur | Legal order of events occurs from one scenario | strings that represent XML code of a diagram of BP Flow syntax | Two events:  “Hello”, “World” | “Hello”, “World” | Pass |
| RandomOrder:  Checks that randomization of requests events occurrence from two scenarios that starting from two different start-nodes is legal. in addition, checks that the probability to choose execute of the same sequence of requests in 100 runs is under 45% (from six legal requests sequence) | Legal order of events occurs from more than one scenarios | strings that represent XML code of a diagram of BP Flow syntax | Legal order of 4 events:  "1", "2", "3", "4"  or  "3", "4", "1", "2"  or  "1", "3", "2", "4"  or  "1", "3", "4", "2"  or  "3", "1", "2", "4"  or  "3", "1", "4", "2" | "1", "3", "2", "4" | Pass |
| HotCold:  Checks the program HotCold - Checks the order of event requests that occur in conjunction with block and wait events | Legal order of events occurs from more than one scenarios that include:  Request, block and wait events in bsync nodes | strings that represent XML code of a diagram of BP Flow syntax | Legal order of events:  Hot, Cold, Hot, Cold, Hot, Cold | Hot, Cold, Hot, Cold, Hot, Cold | Pass |
| Payload: Checks that the payloads that are inserted in the start node pass between nodes, and checks the value of them. | Current Payloads pass  between nodes | strings that represent XML code of a diagram of BP Flow syntax | The correct value of the payloads:  [{"x":3},{"y":4}] | [{"x":3},{"y":4}] | Pass |
| PayloadChange:  Checks that the value of the payloads that are inserted in the start node could be changed in a general block. passes the payloads with the new changes between nodes and checks the current new value of them. | Payloads can change in general node | strings that represent XML code of a diagram of BP Flow syntax | The correct value of the payloads after changes of their values:  [{"x":5},{"y":6}] | [{"x":5},{"y":6}] | Pass |
| PayloadsIfElse:  Checks that general node send other payloads to other outputs, according to the user-defined in the "if-else" condition which is found in the code editor of the general node. | General node pass payload to selected exit outputs point | strings that represent XML code of a diagram of BP Flow syntax | The correct value of the payloads that pass from different outputs:  {"x":3} | {"x":3} | Pass |
| IllegalGraph:  check if the graph has a lonely start node or edge without target or source. | Detection of illegal graph elements | strings that represent XML code of a diagram of BP Flow syntax, that include lonely start node and edge without target | List of 2 element:  1- lonely start node  2- edge without target | List of 2 element:  1- lonely start node  2- edge without target | Pass |
| LegalGraph:  check if the graph has a lonely start node or edge without target or source. | Legal graph rules | strings that represent XML code of a diagram of BP Flow syntax, that not include lonely start node or edge without target | Empty list | Empty list | Pass |
| ExceptionHandle:  check that when occur error while executing the JS code on node the execution of the scenario is terminated. | handle error while executing JS code | strings that represent XML code of a diagram of BP Flow syntax, that include JS code in the code editor of the general node that made an exception | One event:  “Before error” | “Before error” | Pass |
| ExceptionHandle2:  check that when occur error while executing the JS code on node the execution of the scenario is terminated and the others scenarios continue to run. | handle error while executing JS code | strings that represent XML code of a diagram of BP Flow syntax, that include 3 scenarios that in one of them has JS code in the code editor of the general node that made an exception | Seven events:  That not include the event “after error” and includes: “Before error”, 1, 2, 3, 4, 5 | 1, 4, Before error, 2, 5, 6, 3 | Pass |

1. Debugging:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name/Description | Goal | Input | Expected Result | Actual Result | Pass/Fail |
| RequestsList: Checks that only one of the requested events that appear in one bsync block have occurred. in 1addition, in addition, checks that the probability to choose one of the requests in 100 runs is under 80% (from two requests). | The right order of selected request from bsync. the correctness of the payloads at each step in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  1.1 | File reference: Debugger testing result  1.2 | Pass |
| HelloWorld: Checks the order of event requests that occur | The right order of events occurs from one scenario. the correctness of the payloads at each step in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  2.1 | File reference: Debugger testing result  2.2 | Pass |
| RandomOrder:  Checks that randomization of requests events occurrence from two scenarios that starting from two different start-nodes is legal. in addition, checks that the probability to choose execute of the same sequence of requests in 100 runs is under 45% (from six legal requests sequence) | The right order of events occurs from more than one scenarios.  the correctness of the payloads at each step in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  3.1 | File reference: Debugger testing result  3.2 | Pass |
| HotCold:  Checks the program HotCold - Checks the order of event requests that occur in conjunction with block and wait events | The right of events occurs from more than one scenarios that include:  Request, block and wait events in bsync nodes.  the correctness of the payloads at each step in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  4.1 | File reference: Debugger testing result  4.2 | Pass |
| Payload: Checks that the payloads that are inserted in the start node pass between nodes, and checks the value of them. | Current Payloads pass  between nodes in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  5.1 | File reference: Debugger testing result  5.2 | Pass |
| PayloadChange:  Checks that the value of the payloads that are inserted in the start node could be changed in a general block. passes the payloads with the new changes between nodes and checks the current new value of them. | Payloads can change in general node  in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  6.1 | File reference: Debugger testing result  6.2 | Pass |
| PayloadsIfElse:  Checks that general node send other payloads to other outputs, according to the user-defined in the "if-else" condition which is found in the code editor of the general node. | General node pass payload to selected exit outputs point  in debug mode. | strings that represent XML code of a diagram of BP Flow syntax | File reference: Debugger testing result  7.1 | File reference: Debugger testing result  7.2 | Pass |