Test Plan for: Risk Assessment Metric Application
Version: 1.0

Scope:

Minimum viable product providing functionality relating to adding metrics and their corresponding risk values, displaying individual statistics/statuses and overall status of the project.

People:

A team of two developers (Brandon Hillbom and Dillan Zurowski) will complete and present their application for No-Risk Software, Inc. The developers will directly report to Adam Tilson for guidance and project specifications

Test Strategy:

- Create documentation plan that will outline the main areas of testing to be implemented.
- Create documentation outline test specifications.
- Implement the main functionality and then use a Test Driven Development approach when fixing bugs and other defects encountered.
- Document a log of important defects that should be flagged for future consideration.
- Retest the application after any changes made to the program.
- Organize tests in an easily understandable format (UI testing in its own test file, functionality testing in another test file, black box testing recorded in documentation).
- Ensure and verify that all paths are covered and each test uses risky values to attempt to find maximum defects.
- Cover test items and barriers to improvement in daily team meetings.
- Update time estimates and plan time allocation accordingly.

Test activities and estimates:

Test effort is expected to take about 40% of the project time. An estimate of 6 hours worth of testing is our current projection with an emphasis on time estimation subject to change. Near the end of the project, widget testing took a considerable amount of time and there was a steep learning curve with undesirable results

Test Design Documentation:

ID	Features to be tested	Pass/fail criteria	Procedure steps

1	Boundary tests	Ensure all boundary values for status are acceptable and process the risk properly	Perform boundary tests: -.01, 0, 0.01, 0.32, 0.33, 0.34, 0.65, 0.66, 0.67, 0.99, 1, 1.01
2	Path coverage	Ensure that individual paths are all being tested	Cover all logic statements and functions
3	Widgets/add button	When the user enters the add button at the bottom, a new widget with the risk assessment requirements should appear below the previous widget.	Perform widget tests as done during the lecture with the inputs/buttons expecting some behaviour after
4	Score	The scores shown should match the scores calculated.	Create 2 widget boxes, calculate expected score and compare with the actual score.

Test Case Specifications:

Test ID	1
Description	Test inputs at all important boundaries
Precondition	Widgets are created by the user and has entered values into the votes section

Step	**Procedure**	**Expected result**
1	Test values +/- 0.1 around 0,⅓, ⅔, and 1	<0 & >1= Unknown >0 & < ½ = RED >½ & < ⅔ = YELLOW >⅔ & <1 = GREEN
2	Test for non integer inputs	Text field should erase and reset

Test ID	2
Description	Path coverage
Precondition	The user is entering values and the app is calling functions

Step	Procedure	Expected result
1	Create mock scenarios calling the functions	Flutter should be abe to find and call all functions successfully
2	Compare expected return results with actual	The expected return should be the same as the actual for each function

---

Test ID	3
Description	Widgets testing
Precondition	Widgets are created by the user and has entered values into the votes section

Step	Technique	Procedure	Expected result
1	Automatic	Open the application	The default metrics should appear automatically
2	User-input	Enter values into the votes	The scores and total risk should be dynamically changing
3	User-input	Click the add button	A new blank widget should appear at the bottom of the page

4	User-input	Add new values	Adding new values should change the overall risk and satus

---

Test ID	4
Description	Verify the scoring calculations are accurate
Precondition	Widgets are created by the user and has entered values into the votes section

Step	**Technique**	**Procedure**	**Expected result**
1	Manual	Grab expected inputs from the excel sheet provided	Capture the expected results
2	User-input	Create 2 widget boxes and enter the same inputs as before	The scores and total risk should be the same as the ones calculated before

Equivalence Class Testing:

Only 1 of the 4 levels of risk needs an inputted value as calculations will be made and taken using 0 as the value for the empty vote fields.

Minimum Vote: X	Minimum Vote: -	Minimum Vote: -	Minimum Vote: -
Low Vote: -	Low Vote: X	Low Vote: -	Low Vote: -
Reasonable Vote: -	Reasonable Vote: -	Reasonable Vote: X	Reasonable Vote: -
High Vote: -	High Vote: -	High Vote: -	High Vote: X

Defect log

ID	**Description**	**Source of issue**	**Fixed**	**Solution**
1	Issue dynamically adding widgets	ListBuilder	Yes	Used .map(<widget>).tolist
2	Page crashing on wrong input type	FormatExceptio n, only int accepted	Yes	If statement checking for type, clears input on incorrect type
3	Widget testing not finding dynamically added widgets	Can't find new Keys	No	N/A
4	Integer values displaying .00 after the value	.toStringAsFixed (2)	Yes	Removed extra code for total votes displayed value

User Acceptance Testing

ID	Objective	Steps	Expected Result	Actual Result	Pass
1	Create a "red status" table	Add a table, fill out the metrics & input min votes: 1, low votes: 1, reasonable votes: 3, high votes:5	status: red, risk: 0.733 , low score: 0.33, med score: 2, high score: 5, total votes: 10, overall status: red, overall risk: 0.733	status: red, risk: 0.733 , low score: 0.33, med score: 2, high score: 5, total votes: 10, overall status: red, overall risk: 0.733	✓
2	Create a "yellow status" table	Add a table, fill out the metrics & input min votes: 5, low votes: 4, reasonable votes: 5, high votes: 4	status: yellow, risk: 0.481, low score: 1.33, med score: 3.33, high score: 4, total votes: 18, overall status: yellow,	status: yellow, risk: 0.481, low score: 1.33, med score: 3.33, high score: 4, total votes: 18, overall status: yellow,	✓

			overall risk: 0.607	overall risk: 0.607	
3	Create a “green status” table	Add a table, fill out the metrics & input min votes: 6, low votes: 2, reasonable votes: 2, high votes: 0	status: green, risk: 0.2, low score: 0.67, med score: 1.33, high score: 0, total votes: 10, overall status: yellow, overall risk: 0.472	status: green, risk: 0.2, low score: 0.67, med score: 1.33, high score: 0, total votes: 10, overall status: yellow, overall risk: 0.472	✓
4	Add an empty table	Add an empty table, ensure the overall status and risk are unchanged.	status: unknown, risk: “please add values”, overall status: yellow, overall risk: 0.472, 0.00 for all scores	status: unknown, risk: “please add values”, overall status: yellow, overall risk: 0.472, 0.00 for all scores	✓
5	Check missing vote input	Add a table and input a vote value for 1 to 3 of the risk levels (equivalence class)	The missing inputs will be taken as 0. No errors.	The missing inputs are taken as 0. No errors.	✓