KIUWAN REPORT

SOFTWARE ANALYTICS REPORT

Application: MobifinTapestryBackend
Analysis label: no label
Analysis date: Aug 4, 2025, 5:00 PM
Report date: Aug 4, 2025, 5:26 PM

kiuwan

PANAMAX

TABLE OF CONTENTS

Included in this report

SOFTWARE EVALUATION METHODOLOGY

checKing Quality Model for Software (CQM) is Kiuwan methodology for evaluating the internal quality of a software product. **CQM is ISO-25000 based.** It defines us the internal quality scope and characteristics. CQM simplifies ISO 25000 focusing on internal quality. CQM proposes indicators for the following software characteristics:

Security. The capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them.

Reliability. The capability of the software product to maintain a specified level of performance.

Efficiency. The capability of the software product to provide appropriate performance relative to the amount of resources used under stated conditions.

Maintainability. The capability of the software product to be modified. Modifications may include corrections, improvements or adaptability of the software to changes in environment and in requirements and functional specifications.

Portability. The capability of the software product to be transferred from one environment to another.

CQM indicators are normalised to represent these regions:

0-30 region. The characteristic pointed to by the indicator is in the RED zone. Improvements are needed.

11 

30-70 region. Represented by YELLOW and means that you have to keep your mind on this indicator. Your next moves will depend on your requirements.

57

70-100 region. The GREEN zone. This is the zone where all indicators must be. No critical defects founded.

87 

This normalisation allows the comparison of different characteristics between them; this means that you can say if the software is more maintainable than it is efficient or reliable. You are going to compare different version of the same application over time because the meaning of the indicator does not change. You can even compare two different technology applications.

APPLICATION & ANALYSIS INFORMATION

This software has been scanned and vulnerabilities checked for based on the following standards: OWASP, CWE, SANS 25, PCI-DSS, HIPAA, WASC, MISRA-C, BIZEC, ISO 25000, ISO 9126, CERT-C, CERT-J. Kiuwan has analyzed for you a set of source files. In order to put this analysis and your results in context, here are some statistics:

Notes:

- unparsed files are the source code files of your application that kiuwan engine couldn't read during the analysis process.
- unrecognized files are the ones that were in your analyzed path but they aren't source code or they contains code in languages that Kiuwan doesn't support yet.

Analysis Info

label	no label
date	2025/08/04 17:00
ordered by	Nikunj S Darji
encoding	UTF-8
analyzed path	-
languages found	javascript
analyzed files	373
unparsed files	0
unrecognized files	-

Application Info

name	MobifinTapestryBackend
Bussiness value	critical
MFS4x	PORTFOLIO_UNCLASSIFIED
MFS5x	MFS5x
BVI	PORTFOLIO_UNCLASSIFIED
IBMB Apps	PORTFOLIO_UNCLASSIFIED
cms_group	PORTFOLIO_UNCLASSIFIED
Times analyzed	2

Model Info

model name	CQM
model version	q13496
engine version	master.p695.q13496.a1914.i666
active rules	160
active metrics	0

RISK INDEX

Risk index represents the potential problems that you are assuming for not paying attention to the quality of your source code. So far as you are (measured in effort) to get an acceptable quality level.

It is a number that concentrates all the evidence found in the source code of your application. It has been used your global indicator and the effort that you need to spend to reach the quality level set as goal for you. So if you have poor quality, but if the effort needed to get better is low you are not assuming a high risk in this application because you are going to repair your problems easily. But if your effort needed to get better is very high your risk index will be high too.

Pay attention to risk index evolution in time.

RISK INDEX

0

Risk evolution

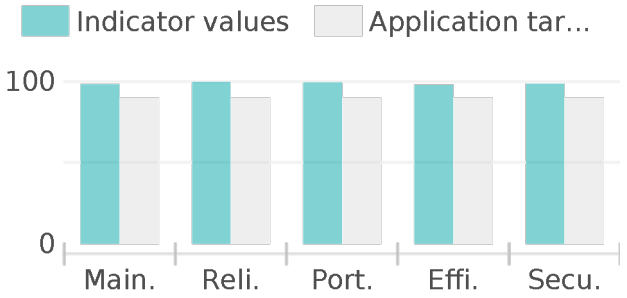
MAIN INDICATORS

You get your global indicator at the right and a breakdown in software characteristics.

Pay attention to the **target values** that have been set when you configure your application. It is important that you know if your quality is better or not that these values and you modify them according to your requirements.

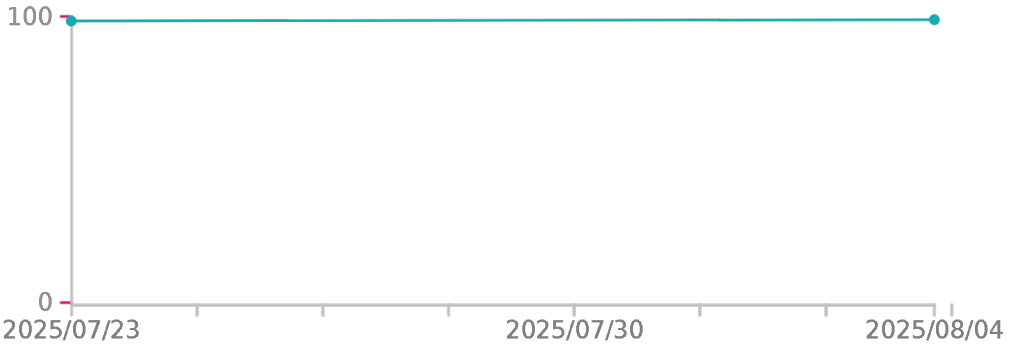
The quality evolution is another important point. **Are you improving?**

GLOBAL INDICATOR
98.96



Characteristic	Indicator	App. target
Maintainability	98.46	90
Reliability	99.99	90
Portability	99.59	90
Efficiency	98.08	90
Security	98.82	90

Global indicator evolution



REPARATION EFFORTS

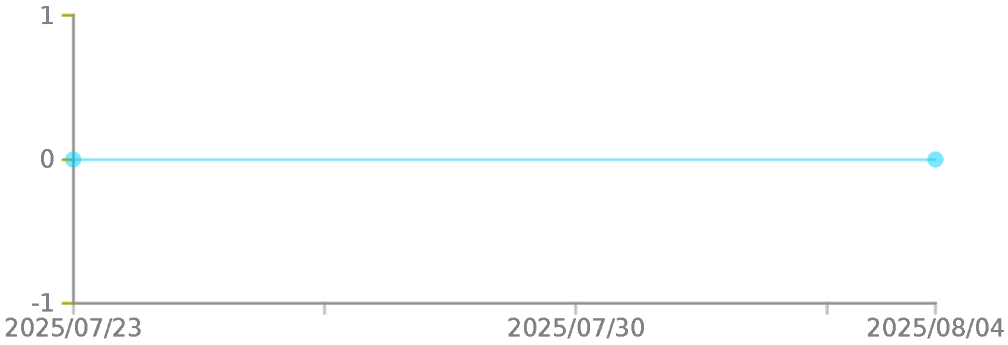
Now you know your quality level, you'll want to know how much it will cost to reach your goal.

We've calculated for you the minimum set of defects will be corrected to achieve it. Here's what you need to invest. You can configure (for accuracy) the effort needed to correct each defect type.

EFFORT TO TARGET
0h

Characteristic	Effort to target
Maintainability	0 h
Reliability	0 h
Portability	0 h
Efficiency	0 h
Security	0 h

Effort to target evolution



MAIN METRIC VALUES

We've computed some metrics of your source code. Below are shown the most important ones.

Lines of code. Excluding commented lines and blank lines.

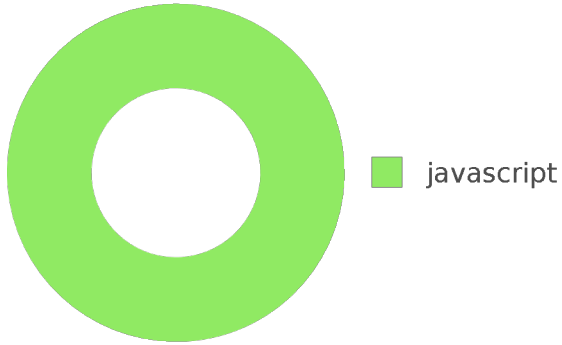
Function points. Functional size calculated by backfiring strategy.

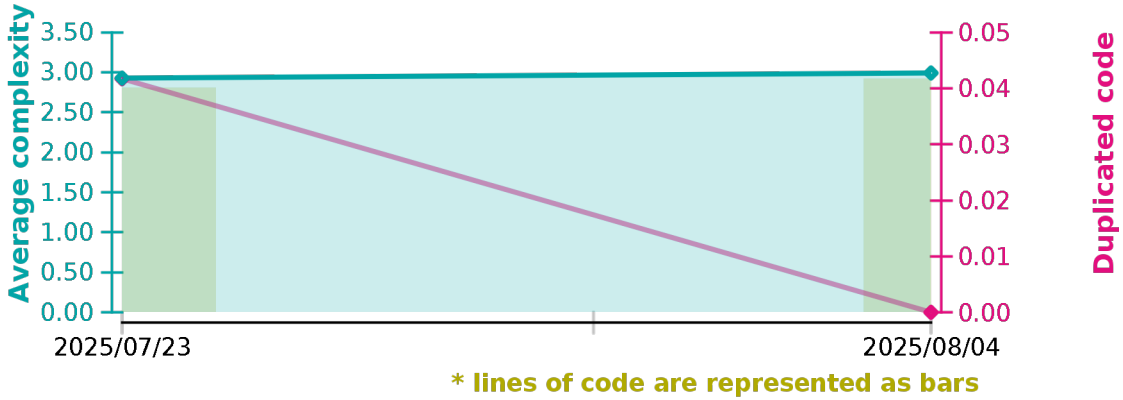
Avg complexity. Average of each function (or method).

Dup code. The ratio of the duplicated code.

Lines of code	46,753
Function points	882.13
Average complexity	2.99
Duplicated code	0%

Lines of code by technology





INDICATOR DISTRIBUTION IN FILES

It's important to know if your issues are distributed homogeneously through files. In the chart we've used **quintiles** dividing the file indicator value in five equal sized subsets, and then we tell you how many files fall in each subset.

In the table, you have a list of the files (the worst ones) that have reduce their indicator from previous analysis. If this is your first analysis, you will get the files with the lower indicator values.

File name	Indicators	Previous	Delta
.../accessControlMiddleware.js	91.17	99.02	-7.85
.../applicationBusinessRoleDTO.js	80.28	80.5	-0.22
.../applicationService.js	93.67	93.87	-0.2
.../foreignKeyValidator.js	99.83	99.96	-0.13
.../globalDataProfileRepository.js	99.93	99.99	-0.05
.../oauthModel.js	99.99	100	-0.01
.../importConfigHistoryRepository.js	99.97	99.98	-0.01
.../globalDataProfile.js	99.9	99.91	-0.01
.../entityAccessRight.js	99.97	99.97	-0.01
.../globalDataProfileDetail.js	99.9	99.91	-0.01
.../workFlowDTO.js	100	100	-0
.../applicationDTO.js	100	100	-0
.../suggestionDTO.js	100	100	-0
.../errorConstant.js	100	100	-0
.../applicationBusinessStructureDTO.js	100	100	-0
.../fileUploadService.js	100	100	-0
.../stringifyJSONConverter.js	100	100	0
.../responseFormatter.js	100	100	0
.../maskData.js	100	100	0
.../jsonToExcel.js	100	100	0
.../formattedDateTime.js	100	100	0
.../fileops.js	100	100	0

Global indicator

METRIC DISTRIBUTION IN FILES

It's important to know if your metrics are distributed homogeneously through files. We've used **quintiles** dividing the file level metric values in five equal sized subsets, and then we tell you how many files fall in each subset.

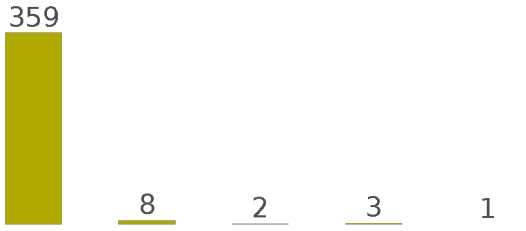
We give you the most three popular metrics:

Size as lines of code of each file.

Complexity as average cyclomatic complexity of each function per file.

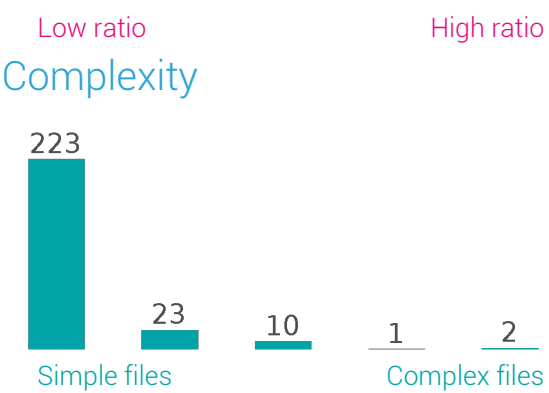
Dup code as the ratio of the duplicated code of each file.

Size



Duplicated code

Complexity



TOP 10 REPAIR FIRST DEFECTS

Here you have a list of top 10 defect types that you have in your application. These defects are the ones that once eliminated give more benefit by unit time of effort. If you want to start fixing quality issues, you must start with this to maximize your time.

Rank	Defects	Files	Rule name	Lang.	Characteristic	Priority	Effort
1	5	1	Avoid errors in the increment or decrement of a variable	javascript	Maintainability	1	30m
2	1	1	Potential denial-of-service attack through malicious regular expression (ReDoS)	javascript	Security	1	6m
3	3	2	External Control of File Name or Path	javascript	Security	1	18m
4	1	1	HTTP parameter pollution (HPP)	javascript	Security	1	6m
5	16	12	Avoid unused local variable	javascript	Maintainability	2	48m
6	1	1	Avoid duplicating property names in object literals	javascript	Reliability	2	3m
7	1	1	Prevent MIME sniffing	javascript	Security	2	3m
8	6	2	Avoid declaring a variable with a name that is already used	javascript	Reliability	2	36m
9	1	1	No clickjacking protection configured	javascript	Security	2	6m
10	64	10	Avoid object instantiation into loops	javascript	Efficiency	2	6h 24

If you want a complete list of defects to repair in order to reach a quality target or to spend a bag of budgeted hours you can take an **action plan** report from kiuwan.com in **what if** function.

KIUWAN REPORT

THANK YOU!

contact@kiuwan.com | Partnership:partners@kiuwan.com | +1 6176073353

kiuwan