Session dates: 11.12.2023.

Reported by: Amar Tahirović

Inspected document: BURGIJA app

Inspection team: BURGIJA

Structure

- Does the code completely and correctly implement the design?
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted? x
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than "magic number" constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

- Is the code clearly and adequately documented with an easy-to-maintain commenting style?
- Are all comments consistent with the code? x

Variables

- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables?

Arithmetic Operations

- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

Loops and Branches

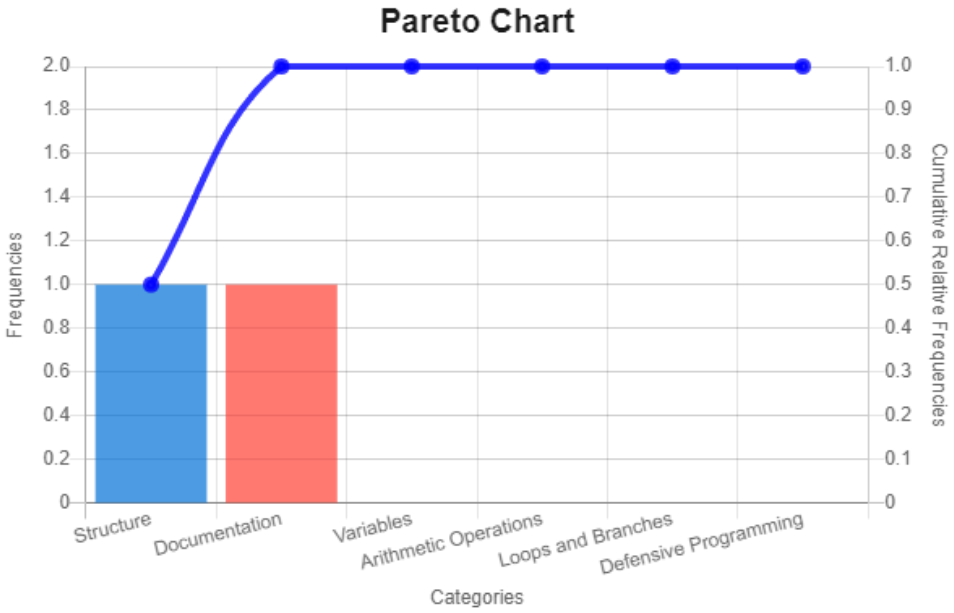
- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

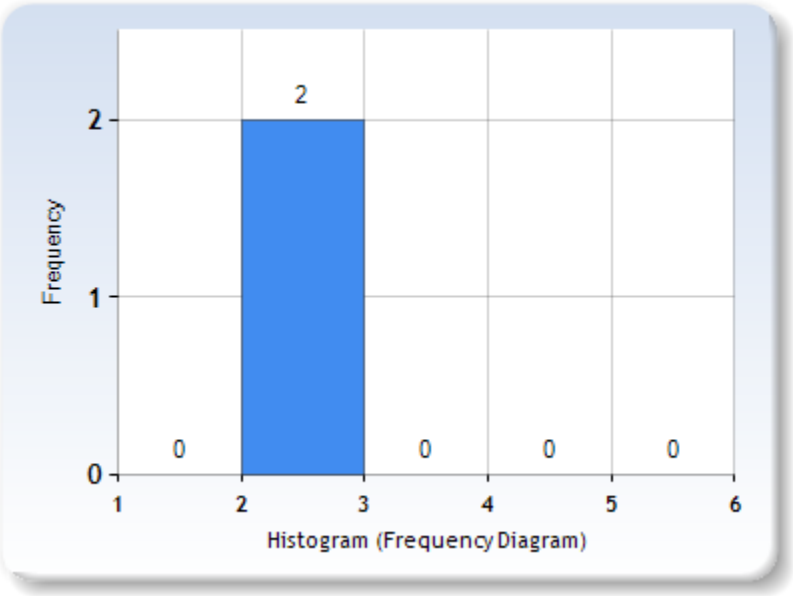
Defensive Programming

- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness?
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

*	Error type	Error nature	Error description	Error location	Error severity
1	Structure	E	Reorganizacijom koda, možemo postići filter zahtjeva pa onda pretragu takvih podataka	Home Controller	2
2	Documentation	M	Nedostatak dokumentacije za Home Controller	Home Controller	2

a	Follow-up will be carried out by: Emir Agović
b	Re-inspection is recommended: Yes





x – osa – ozbiljnost greške
y – osa – broj pronađenih grešaka

Session dates: 12.12.2023.

Reported by: Emin Zukić

Inspected document: BURGIJA app

Inspection team: BURGIJA

Structure
- Does the code completely and correctly implement the design?
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted?
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than "magic number" constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation
- Is the code clearly and adequately documented with an easy-to-maintain commenting style? X
- Are all comments consistent with the code?

Variables
- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables?

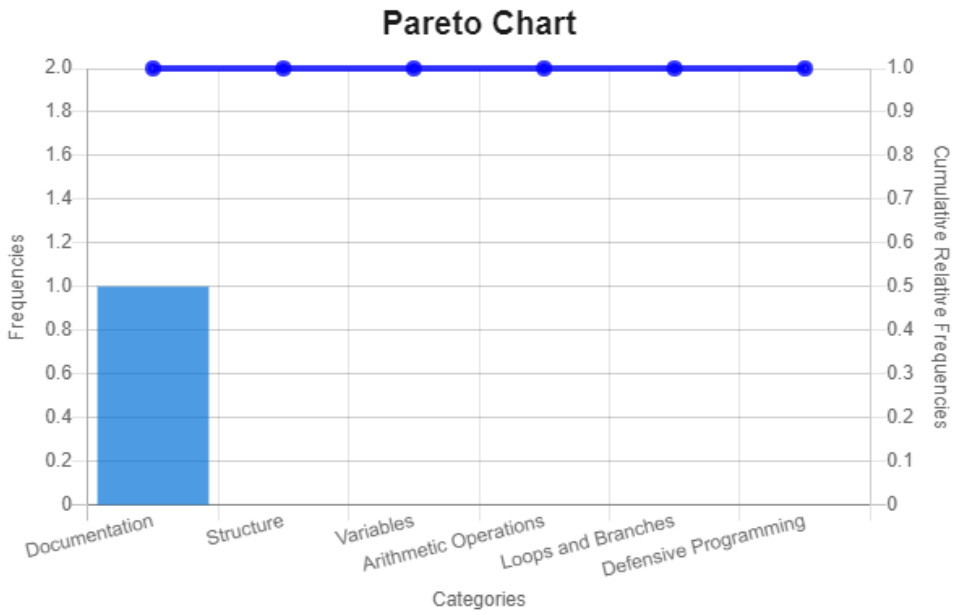
Arithmetic Operations
- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

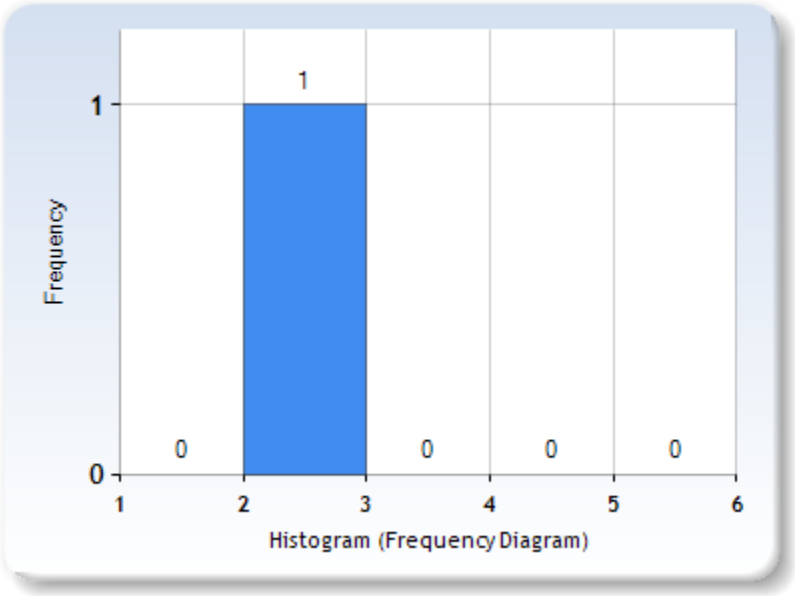
Loops and Branches
- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming
- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness?
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

*	Error type	Error nature	Error description	Error location	Error severity
1	Documentation	M	Nedostatak dokumentacije za AdminPanel Controller	AdminPanel Controller	2

a	Follow-up will be carried out by: Emir Agović
b	Re-inspection is recommended: Yes





x – osa – ozbiljnost greške
y – osa – broj pronađenih grešaka

Session dates: 11.12.2023.

Reported by: Ahmed Ljubunčić

Inspected document: BURGIJA app

Inspection team: BURGIJA

Structure
- Does the code completely and correctly implement the design? X
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted?
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than "magic number" constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation
- Is the code clearly and adequately documented with an easy-to-maintain commenting style? X
- Are all comments consistent with the code? X

Variables
- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables? X

Arithmetic Operations
- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

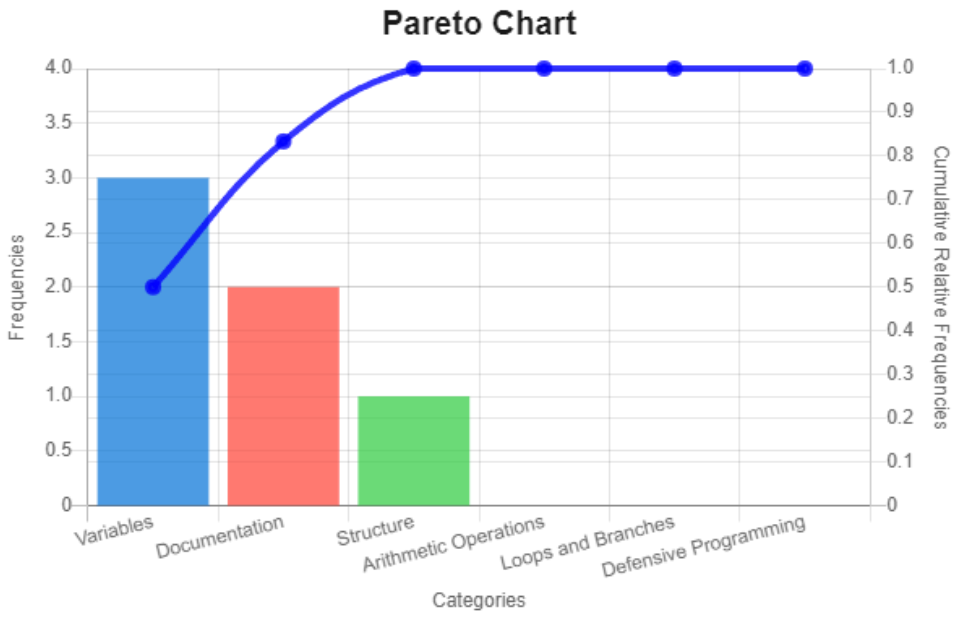
Loops and Branches
- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

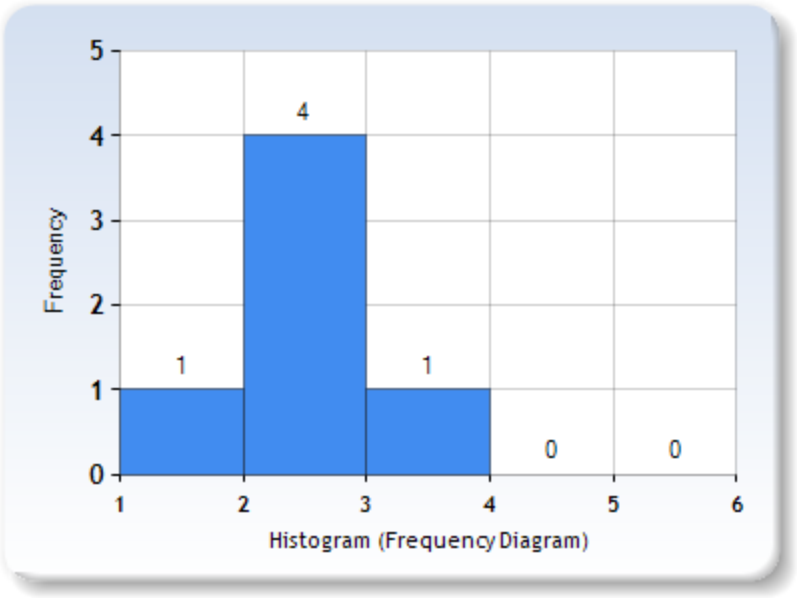
Defensive Programming
- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness?
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

*	Error type	Error nature	Error description	Error location	Error severity
1	Structure	W	Historiju iznajmljivanja bi trebao moći vidjeti samo RegisteredUser, a ne korisnici sa ostalim rolama (Admin i Courier)	RentController (30)	3
2	Documentation	M	Nedostatak dokumentacije u Rent controlleru	RentController (19, 30-31, 42-43, 55-56, 81-86)	2
3	Documentation	M	Kod nije objašnjen kroz komentare	RentController (33-39, 88-117)	2
4	Variables	E	Nepotreban kod, podaci se ne koriste na pogledima	RentController (73-75, 114-116)	2

5	Variables	W	Discount id uvijek je postavljen na null vrijednost kad se kreira Rent objekat	RentController (105)	2
6	Variables	E	_userManager se nigdje ne koristi u klasi, redundantan atribu	RentController (22)	1

a	Follow-up will be carried out by: Emir Agović
b	Re-inspection is recommended: Yes





x – osa – ozbiljnost greške
y – osa – broj pronađenih grešaka

Session dates: 13.12.2023.

Reported by: Emir Agović

Inspected document: BURGIJA app

Inspection team: BURGIJA

Structure

- Does the code completely and correctly implement the design?
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted? X
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than "magic number" constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

- Is the code clearly and adequately documented with an easy-to-maintain commenting style? X
- Are all comments consistent with the code? X

Variables

- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables?

Arithmetic Operations

- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

Loops and Branches

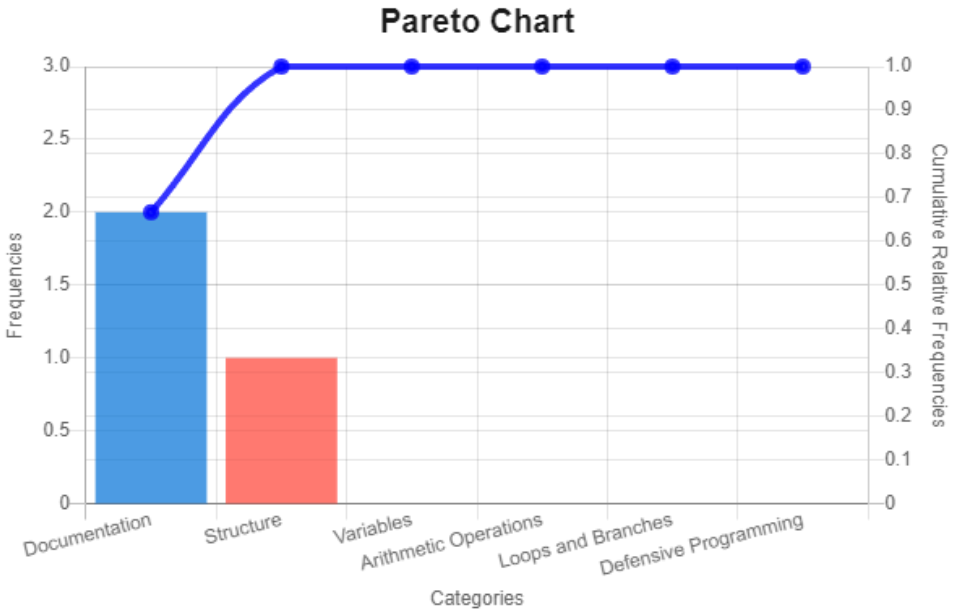
- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

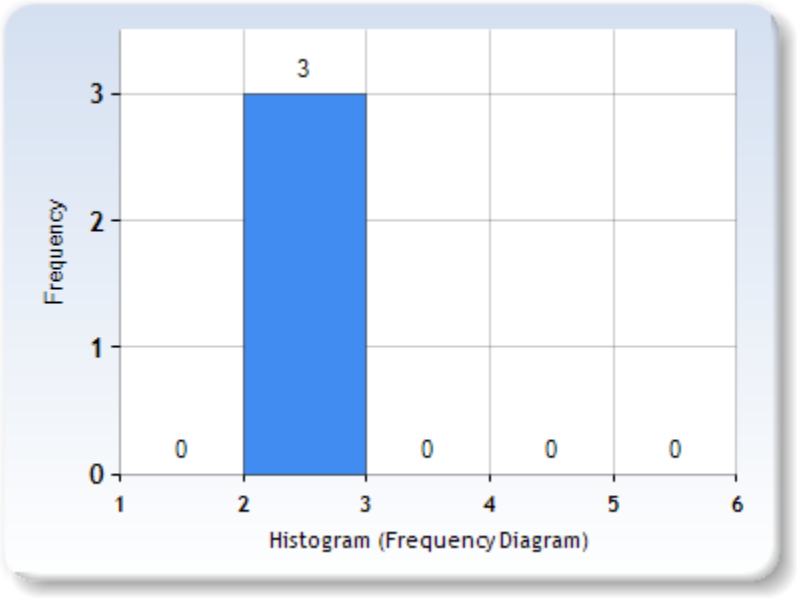
Defensive Programming

- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness?
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

*	Error type	Error nature	Error description	Error location	Error severity
1	Structure	W	Dijelovi koda se mogu smjestiti u zasebne funkcije.	Home Controller	2
2	Documentation	M	Kod nije objašnjen kroz komentare.	Home Controller	2
3	Documentation	M	Nedostatak dokumentacije.	Home Controller	2

a	Follow-up will be carried out by: Emir Agović
b	Re-inspection is recommended: Yes





x – osa – ozbiljnost greške
y – osa – broj pronađenih grešaka

Session dates: 13.12.2023.

Reported by: Emir Bronja

Inspected document: BURGIJA app

Inspection team: BURGIJA

Structure

- Does the code completely and correctly implement the design?
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted?
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than "magic number" constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

- Is the code clearly and adequately documented with an easy-to-maintain commenting style?
- Are all comments consistent with the code?

Variables

- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables?

Arithmetic Operations

- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

Loops and Branches

- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

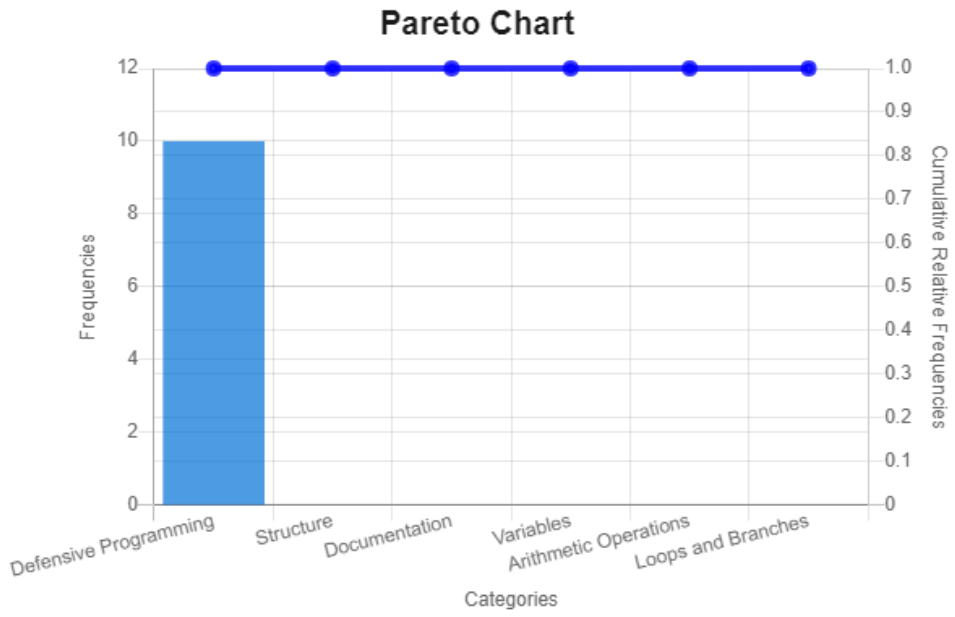
Defensive Programming

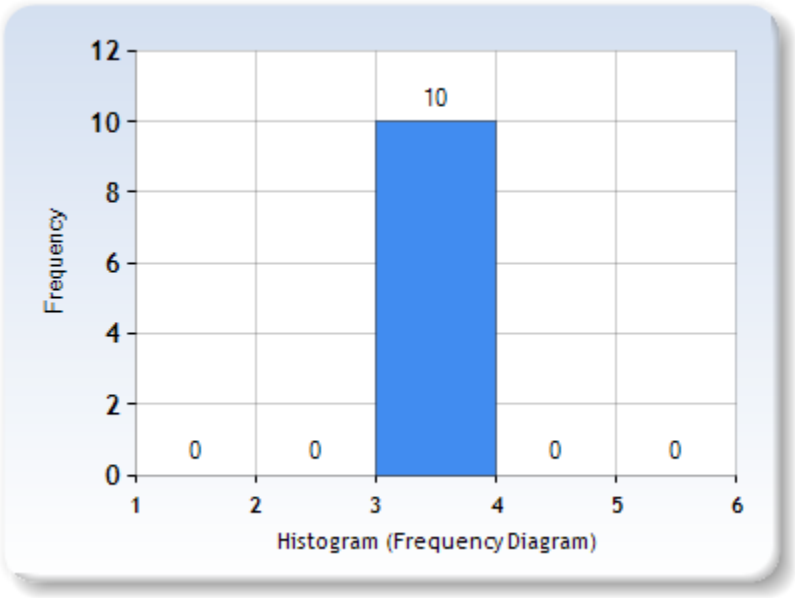
- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness? x
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

*	Error type	Error nature	Error description	Error location	Error severity
1	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Administrator Model	3
2	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Courier Model	3
3	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Delivery Model	3

4	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Location Model	3
5	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	RegisteredUser Model	3
6	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Rent Model	3
7	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Review Model	3
8	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Store Model	3
9	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	Tool Model	3
10	Defensive Progamming	M	Nedostatak validacije proslijeđenih parametara u konstruktorima	ToolType Model	3

a	Follow-up will be carried out by: Emir Agović
b	Re-inspection is recommended: Yes





x – osa – ozbiljnost greške
y – osa – broj pronađenih grešaka