Republic of Rwanda
Ministry of Education

RTB | RWANDA TVET BOARD

DATA STRUCTURE AND ALGORITHM FUNDAMENTALS

SWDDA401

Apply Data Structure and Algorithm Fundamentals Using JavaScript

Competence

RQF Level: 4

Learning Hours 130

Credits: 13

Sector: ICT and Multimedia

Trade: Software Development

Module Type: Specific

Curriculum: ICTSWD4002 -TVET CERTIFICATE IV IN SOFTWARE DEVELOPMENT

Issue Date: September 2023

Purpose statement	This specific module describes the knowledge, skills and attitude required to apply Data Structure and Algorithm Fundamentals using JavaScript. Up on completion of this module, the learner will be able to Apply Algorithm Fundamentals, Apply Data Structure and Implement Algorithm using JavaScript.					
Learning assumed to be in place	▪ Applied Mathematics					
Delivery modality	**Training delivery**		**100%**	**Assessment**		**Total 100%**
	Theoretical content		30%	Formative assessment	30%	50%
	Practical work:		70%		70%	
	Group project and presentation	20%				
	Individual project /Work	50%				
			Summative Assessment			50%

Elements of Competence and Performance Criteria

Elements of competence	Performance criteria
1.Apply Algorithm Fundamentals	1.1 Number systems are correctly converted according to the base conversion methods
	1.2 Logic gates and expressions are well described based on Boolean algebra

	1.3 Data types are effectively used according to their intended use
	1.4 Operators are appropriately used based on datatype
	1.5 Algorithm is properly written based on problem to be solved
2.Apply Data Structure	2.1 Data structure concepts are clearly identified based on intended use.
	2.2 Linear Data Structures are properly applied based on their operational complexity
	2.3 Non-Linear Data Structures are properly applied based on their operational complexity
3.Implement Algorithm using JavaScript	3.1 JavaScript Source code is properly written based on Algorithm
	3.2 JavaScript source code is successfully run in accordance with expected result
	3.3 Time and space complexity are successfully tested based on data structure standards

Course content

Learning outcomes	At the end of the module the learner will be able to: 1. Apply Algorithm Fundamentals 2. Apply Data Structure 3. Implement Algorithm using JavaScript

Learning outcome 1: Apply Algorithm Fundamentals	Learning hours: 30
Indicative content	

- **Conversion of number systems**
  - ✓ Description of key concepts
    - Decimal base
    - Binary base
    - Hexadecimal base
    - Octal base
    - Unary encoding
  - ✓ Number system from decimal base to:
    - Binary base and vice versa
    - Octal base and vice versa
    - Hexadecimal and vice versa
  - ✓ Number system from hexadecimal base to:
    - Binary base and vice versa
    - Octal base and vice versa
    - Decimal and vice versa
  - ✓ Number system from base Octal base to:
    - Binary base and vice versa

- Decimal base and vice versa
- Hexadecimal base and vice versa

✓ Application of number base arithmetic operations

- **Description of logic gates and expressions**

✓ Representation of Boolean logic gates

- AND gate
- NAND gate
- OR gate
- NOR gate
- XOR gate

✓ Application of Boolean logic gates

- Circuits
- Truth table

- **Use of data types on variables**

✓ Definition of datatype

✓ Data types used in JavaScript

- Primitive data types
- Non-Primitive data types

✓ Application of datatypes

- **Application of JavaScript operators**

✓ Assignment operators

✓ Arithmetic operators

✓ Logical operators

✓ Relational operators

✓ Compound operators

✓ Conditional operators

✓ Bitwise operators

- **Write an algorithm**

✓ Definition

- ✓ Types of algorithm
- ✓ Characteristics/qualities of a good algorithm
- ✓ Develop an algorithm using structured English
- ✓ Develop an algorithm using pseudocode
  - Sequence structures
  - Selection/conditional structures
  - Looping/iterating structures
- ✓ Design of Flowchart
  - Description of Elements of Flowchart
  - Using Flowchart tools
  - Apply Flowchart best practices
- ✓ Draw a flowchart

Resources required for the learning outcome	
Equipment	▪ Computer
Materials	▪ Internet ▪ Papers ▪ Pencils ▪ Electricity ▪ Training manual
Tools	▪ Visual paradigm ▪ Edraw max ▪ Lucidchart
Facilitation techniques	▪ Practical exercise ▪ Trainer guided ▪ Group discussion ▪ Demonstration ▪ Individual practical exercise

Formative assessment methods /(CAT)	▪ Written assessment ▪ Performance assessment

Learning outcome 2: Apply Data Structure	**Learning hours: 45**
Indicative content	

• **Identification of data structure concepts**
✓ Definition
✓ Classifications of data structures
  ➕ Linear
  ➕ Non-linear
✓ List representation
✓ List operations
✓ Structure
✓ Searching techniques
  ➕ Binary search
  ➕ Linear search
✓ Time complexity
✓ Space complexity
✓ Classification of sorting algorithms
  ➕ By number of comparisons
  ➕ By Number of Swaps
  ➕ By Memory Usage
  ➕ By Recursion
  ➕ By Stability
  ➕ By Adaptability
  ➕ Internal Sorting	

- External Sorting

✓ Sorting techniques

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Shell Sort
- Heap Sort
- Radix Sort
- Counting Sort
- Bucket Sort

- **Application of linear data structures and their operations**

✓ Linked lists

✓ Arrays

✓ Queue

✓ Stack

✓ Write procedures

- **Application of non-linear data structure and their operations**

✓ Tree

✓ Graph

✓ Tables

✓ Write procedures

Resources required for the indicative content	
Equipment	▪ Computer
Materials	▪ Internet ▪ Training manual

	▪ Electricity
Tools	▪ Python tutor ▪ VisuAlgo
Facilitation techniques	▪ Demonstration ▪ Trainer guided ▪ Group discussion ▪ Individual practical exercise
Formative assessment methods /(CAT)	▪ Written assessment ▪ Performance assessment

Learning outcome 3: Implement Algorithm using JavaScript	**Learning hours: 55**
Indicative content	
• **Development of JavaScript source code** ✓ Preparation of JavaScript running environment ✓ Writing JavaScript source code Linked lists Arrays Queue Stack Tree Graph Tables ✓ Perform sorting operations	

- Bubble
- Quick

✓ Perform searching operations

- Binary
- Linear

- **Run JavaScript source codes**

✓ Using browser embedded Tools

- Rendering engine
- Web dev tools

✓ Using IDE Terminal

- **Test Time and space complexity**

✓ Key concepts of measuring time and space complexity

✓ Time and space measurement tools

- Profiling tools
- Benchmark.js
- Benchmarkify
- jsPerf

✓ Document test findings

Resources required for the indicative content	
Equipment	▪ Computer
Materials	▪ Internet ▪ Training manual ▪ Electricity
Tools	▪ Benchmarkify.js ▪ jsPerf ▪ WebStorm

	▪ Visual Studio Code IDE built-in profiling tool ▪ Frame graphs ▪ Blackfire ▪ YSlow ▪ Chrome/Microsoft/Firefox DevTools
Facilitation techniques	▪ Brainstorming ▪ Trainer guided ▪ Group Discussion ▪ Individual Practical exercise
Formative assessment methods /(CAT)	▪ Written assessment ▪ Performance

Integrated/Summative assessment

Integrated Situation

SmartPark is a car parking management company located in Rubavu District. Normally, its working principle is organized in a way that each incoming car is given a token having car number plate, parking slot, date, and time of entrance. They charge the car driver an amount of Rwf500/hour for parking service during exit. Once the parking time exceeds, the extra time is charged 300Rfrw per extra hour. All these processes are done manually using paper-based system.

This leads to the various challenges including:

- Difficulties in finding out the actual status of the car park quickly to allow or deny a new incoming car to park.
- Inaccuracy in setting time-in and time-out for incoming car
- The car park manager is exposed to easily be mistaken when recording time spent by the car on a parking slot.
- It takes a time to calculate total amount to be paid by the car drivers at exit.

Therefore, the management has decided to make improvement by computerizing the parking management system by hiring you as a skilled developer in Data Structure and Algorithm to develop: (1) the procedure and flowchart, (2) develop source code using JavaScript which will provide solutions by automating the old system.

TASKS:

(i) Create the car park with the maximum space for 50 cars.

(ii) Check whether the car park is full

(iii) Enter the cars in the car park

(iv) Remove cars from the car park and update its status

(v) Traverse all the cars within the car park.

(vi) Document time and space complexity test findings.

INSTRUCTIONS:

1. Use the idea of array data structure.
2. The procedure should be JavaScript-oriented and submitted.
3. Flowchart should be neatly drawn and submitted.
4. The tasks should be performed within 6 hours.

Resources

Equipment	▪ Computer ▪ Mathematical Set
Materials/ Consumables	▪ Electricity ▪ Papers ▪ Pen
Tools	▪ Vs Code ▪ Google Chrome ▪ Chrome Dev Tool

Assessable outcomes	Assessment criteria (Based on performance criteria)	Indicator	Observation		Marks allocation
			Yes	No	
Learning Outcome 1: Apply Algorithm Fundamentals **(30%)**	1. Data types are effectively used according to their intended use	Ind 1. Data types are applied on variable			3
		Ind 1. Arithmetic operations are applied in the parking process			2

	2. Operators are appropriately used based on datatype	Ind 2. Relational operations are applied in Boolean expression			2
	3. Algorithm is properly written based on problem to be solved	Ind 1. Iterations are applied			4
		Ind 2. Appropriate symbols are used on flowchart			4
		Ind 3. Logical flow of data is shown			2
		Ind 4. Flowchart has finiteness			2
Learning Outcome 2: Apply Data Structure **(40%)**	1. Linear Data Structures are properly applied based on their operational complexity	Ind 1. Maximum slots variable is initialized to 50			2
		Ind 2. Parked cars array is initialized to null			3
		Ind 3. Parking slots status is checked			5
Learning Outcome 3: Implement Algorithm using JavaScript **(30%)**	1. JavaScript Source code is properly developed based on Algorithm	Ind 1. Fifty parking slots are declared and initialised			5
		Ind 2. Parked cars array is initialised to null			3
		Ind 3. Conditional structures are applied			3
		Ind 4. Operators are applied			2

		Ind 5. Iterations are performed on			5
	2. JavaScript source code is successfully run in accordance with expected result.	Ind 1. Free parking slots status is checked			4
		Ind 2. Incoming car is added into free parking slot.			5
		Ind 3. Occupied slots status is updated			5
		Ind 4. Exiting car is removed from parking slot.			5
		Ind 5. Free slots status is updated			5
		Ind 6. Exiting car is charged Fees based on time spent in parking slot.			5
	3. Time and space complexity are successfully tested based on data structure standards	Ind 1. Time and space complexity test findings is documented.			4
Total marks					**80**
Percentage Weightage					**100%**
Minimum Passing line % (Aggregate): 70%					

References:

1. Bagde, P. (2023, 5 6). *javascriptDocuments*. Retrieved from github.com/priya42bagde: https://github.com/priya42bagde/javascriptDocuments/find/main

2. Kime, M. M. (1984). *Logical and computer Design Fundamentals* (6 ed.). (Pearson, Ed.) Prentice Hall. Retrieved 5 6, 2023, from https://drive.uqu.edu.sa/_/mskhayat/files/MySubjects/20178FS%20LogicAnalysisAndDesign/Logic_and_Computer_Design_Fundamentals_-_4th_International_Edition.pdf

3. Knuth, D. E. (1973). *The Art of Computer Programming Volume 3* (2 ed.). Reading-USA: Addison-Wesley. Retrieved 5 6, 2023, from https://ia601506.us.archive.org/34/items/B-001-001-250/B-001-001-250.pdf

4. McMillan, M. (2015). *Data Structres and Algorithms with JavaScript* (2 ed.). (B. M. Blanchette, Ed.) USA: "O'Reilly Media,. Retrieved from https://books.google.rw/books?id=1ywEAwAAQBAJ&lpg=PP1&pg=PP1#v=onepage&q&f=true

5. tutorialspoint. (2023, 5 6). *number system conversion*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/computer_logical_organization/number_system_conversion

6. V. Karavirta and C. A. Shaffer vol. 9, n. 2.-1.-J. (2016). *"Creating Engaging Online Learning Material with the JSAV JavaScript Algorithm Visualization Library," in IEEE Transactions on Learning Technologies,* (Vol. 9). IEEE. Retrieved 5 6, 2023, from https://ieeexplore.ieee.org/abstract/document/7298430