ourdiu.com | Shared by MD. RUHUL AMIN

Daffodil International University

Faculty of Science & Information Technology

Department of Computer Science & Engineering

Final Examination, Summer 2026

Course Code: CSE442 , Course Title: Software Quality Assurance

Level: 4 Term: 2 Batch: 64

Time: 02:00 Hrs **Marks: 40**

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	***a)***	Consider the following Python function designed to calculate parking fees for an "QuickPark" Automated Garage System:
		`def calculate_parking_fee(duration_minutes, is_registered_member):`
		`  if duration_minutes <= 15:`
		`  fee = 0`
		`  elif duration_minutes <= 60:`
		`  if is_registered_member:`
		`  fee = 0`
		`  else:`
		`  fee = 5`
		`  else:`
		`  fee = 20`
		`  return fee`
		i. Build the Control Flow Graph (CFG) for the code segment above. Number each decision node clearly.
		ii. Determine the Cyclomatic Complexity V(G). List all linearly independent Test Paths along with specific input values (duration_minutes, is_registered_member) required to execute each path.
			5	**CO2**

	b)	Consider the "QuickPark" Automated Garage Access Module with the following lifecycle: When a driver approaches the gate and requests entry, the system enters the REQUESTED state. When the system verifies a valid account and credit card, it transitions to GRANTED.From GRANTED, if the vehicle backs away before entering the garage, it moves to ABANDONED (a final state). Once the vehicle crosses the sensor and enters the garage, it transitions to PARKED.When the vehicle approaches the exit gate and payment is successfully processed, it transitions to EXITED (a final state). i. Build the Finite State Machine (FSM) diagram showing all States, Transition Events, and resulting Actions. ii. Determine the corresponding State Transition Table detailing: *Current State*, *Input Event*, *Next State*, and *Output Action*.	**5**	

ourdiu.com | Shared by MD. RUHUL AMIN

2.	Client Requirement Notes of "QuickPark" Automated Garage System.**:** "Drivers must have a registered account and a valid credit card on file to enter the garage. Standard parking costs $5.00 per hour, but the first 15 minutes are always free. Additionally, garage administrators need an 'Emergency Gate Override' button on their control dashboard, but clicking it must display a secondary confirmation modal requiring an admin PIN before opening the gate."			**CO3**
	a)	Develop FOUR (4) distinct, formal, numbered requirements (REQ-01 to REQ-04) from the client notes above.	**4**	
	b)	Construct FOUR (4) high-level Test Scenarios (TS-01 through TS-04) covering all extracted requirements. Then construct a Requirements Traceability Matrix (RTM) table.	**4**	
	c)	Select REQ-01 (Account & Card Validation) and REQ-02 ($5/hr Rate & 15-Min Free Rule). List TWO (2) standard, detailed Test Cases (1 Positive Path and 1 Negative Path for each).	**8**	
	d)	Distinguish each of the following 4 independent software bugs logged during testing into its correct Severity and Priority level combination. Provide a 1-sentence justification for each choice. Bug 1: The company logo on the printed garage exit receipt appears blurry and low-resolution. Bug 2: The primary 'Pay & Exit' button text is misspelled as "Pya & Exit" on the main driver kiosk screen seen by all customers. Bug 3: The system crashes and corrupts database logs when an administrator enters a 255-character special character string in the search bar at 11:59 PM. Bug 4: The payment gateway freezes during processing and charges double the parking fee on all credit card transactions across the system.	**4**	

3.	***a)***	As a QA engineer, you receive this JSON response when an administrator triggers a manual override in the garage system:
		`{`
		`  "garageId": "GRG-772",`
		`  "gateId": "GATE-04",`
		`  "action": "EMERGENCY_OVERRIDE",`
		`  "requestedAt": "2026-07-28T10:20:00Z",`
		`  "adminEmail": "admin@admin.com",`
		`  "tariffRulesApplied": null`
		`}`
		Analyze the payload and list four distinct structural or formatting checks a tester must perform to verify this response matches its API contract.
			5	**CO4**

	b)	A driver wants to enter the garage but does not have a valid credit card saved on their profile. Develop a Playwright test script in Python that navigates to the garage access page, attempts to click the entry button, and verifies that the system blocks entry and prompts them to add a payment method.	**5**	