Method	Test ID	Parameter Values	Comment	Expected Outcome
itemAvailability(itemID:string) : catalogueItem	1	itemID = "C001"	Item exists in catalogue and is in stock.	“success” (a non-null CatalogueItem is returned). Returned catalogueItem has correct details for C001 and quantity > 0.
itemAvailability(itemID:string) : catalogueItem	2	itemID = "C003"	Item exists in catalogue but is out of stock (quantity = 0).	“success” (a non-null catalogueItem is returned). Returned catalogueItem has correct details for C003 and quantity = 0.
itemAvailability(itemID:string) : catalogueItem	3	itemID = "UNKNOWN_ITEM"	Item does not exist in catalogue.	Failure (null returned). No stock is changed.
itemAvailability(itemID:string) : catalogueItem	4	itemID = "" (empty)	Empty itemID sent.	Failure (null returned). No stock is changed.
searchStock(keyword:string) : List<catalogueItem>	5	keyword = "para"	Keyword matches existing items (e.g., “Paracetamol”).	“success” (non-empty List returned). List contains only matching catalogueItem entries.
searchStock(keyword:string) : List<catalogueItem>	6	keyword = "zzzz"	Keyword matches no items.	“success” (empty List returned). No stock is changed.

searchStock(keyword:string) : List<catalogueItem>	7	keyword = "" (empty)	Empty keyword sent.	Failure (empty List returned or error indicator, depending on implementation rule). No stock is changed.
searchStock(keyword:string) : List<catalogueItem>	8	keyword = "PARA"	Case-insensitive search check.	“success” (same results as keyword = “para”). No stock is changed.
deductStock(itemID:string) : boolean	9	itemID = "C001"	Item exists and quantity > 0.	“success” (Boolean returned is “true”). Quantity reduced by 1.
deductStock(itemID:string) : boolean	10	itemID = "C003"	Item exists but quantity = 0.	Failure (Boolean returned is “false”). Quantity remains 0.
deductStock(itemID:string) : boolean	11	itemID = "UNKNOWN"	Item does not exist.	Failure (Boolean returned is “false”). No stock changed.
deductStock(itemID:string) : boolean	12	itemID = ""	Empty item ID sent.	Failure (Boolean returned is “false”). No stock changed.

getCatalogue() : List<catalogueItem>	13	(none)	Catalogue contains items.	“success”. Non-null List returned with one or more catalogueItem objects.
getCatalogue() : List<catalogueItem>	14	(none)	Catalogue empty scenario.	“success”. Empty List returned (not null).
StockManager(dbConnection:string)	SM-1	dbConnection = "jdbc:valid-connection-string"	Construct StockManager with a valid DB connection string.	“success”. StockManager object created. dbConnection stored correctly. No connection attempted until connectToDatabase() is called (unless your implementation connects in constructor).
StockManager(dbConnection:string)	SM-2	dbConnection = "" (empty)	Construct with empty connection string.	Failure behaviour defined (either constructor rejects and throws error, or StockManager created but connectToDatabase will fail). No database connection established.

connectToDatabase():void	SM-3	(none)	Valid dbConnection has been configured; database reachable.	“success”. Connection established (e.g., no exception; internal connected state true). Subsequent updateLocalStorage() succeeds.
connectToDatabase():void	SM-4	(none)	dbConnection invalid/unreachable.	Failure. Connection not established (exception handled or error state set). No data loaded into localDatabase.
updateLocalStorage():void	SM-5	(none)	Database contains items; localDatabase should be refreshed from DB.	“success”. localDatabase populated/updated to match database values (same item IDs and quantities).
updateLocalStorage():void	SM-6	(none)	Database is empty.	“success”. localDatabase becomes empty (or remains empty). No exception.
updateLocalStorage():void	SM-7	(none)	Called when not connected to database.	localDatabase not updated. Error reported/handled (no crash).
updateDatabase():void	SM-8	(none)	localDatabase contains changes that should be written back to DB.	“success”. Database updated to match localDatabase (same IDs and quantities).

updateDatabase():void	SM-9	(none)	Called when not connected to database.	Failure. Database not updated. Error reported/handled (no crash).
updateDatabase():void	SM-10	(none)	localDatabase contains invalid quantity (e.g., negative) for an item.	Failure. Invalid values rejected; database not updated with negative stock. Error reported/handled.
itemAvailability(itemID:string):catalogueItem	SM-11	itemID = "C001"	Item exists in localDatabase (quantity > 0) after updateLocalStorage().	“success”. Returns catalogueItem for C001 with quantity matching localDatabase.
itemAvailability(itemID:string):catalogueItem	SM-12	itemID = "C003"	Item exists but quantity = 0 in localDatabase.	“success”. Returns catalogueItem for C003 with quantity = 0.
itemAvailability(itemID:string):catalogueItem	SM-13	itemID = "UNKNOWN"	Item does not exist in localDatabase.	Failure. Returns null. No changes.
itemAvailability(itemID:string):catalogueItem	SM-14	itemID = ""	Empty itemID sent.	Failure. Returns null. No changes.
searchStock(keyword:string):List<catalogueItem>	SM-15	keyword = "para"	localDatabase/catalogue contains matching item names.	“success”. Non-empty list returned, matches only.
searchStock(keyword:string):List<catalogueItem>	SM-16	keyword = "zzzz"	No matches.	“success”. Empty list returned (not null).

searchStock(keyword:string):List<catalogueItem>	SM-17	keyword = ""	Empty keyword sent.	Failure or empty list returned (depending on your rule). No changes.
deductStock(itemID:string):boolean	SM-18	itemID = "C001"	Item exists and quantity > 0 in localDatabase.	“success” (true). localDatabase quantity for C001 decreases by 1. If implementation persists immediately, DB also updated; otherwise change remains local until updateDatabase().
deductStock(itemID:string):boolean	SM-19	itemID = "C003"	Item exists but quantity = 0.	Failure (false). Quantity remains 0 (no negative stock allowed).
deductStock(itemID:string):boolean	SM-20	itemID = "UNKNOWN"	Item does not exist.	Failure (false). No changes.
deductStock(itemID:string):boolean	SM-21	itemID = ""	Empty itemID sent.	Failure (false). No changes.
deductStock(itemID:string):boolean	SM-22	itemID = "C010" (quantity becomes exactly lowStockThreshold after deduction)	Boundary case for low stock threshold.	“success” (true). Quantity reduced. Low-stock condition triggered only if your rule is “< threshold” (so at exactly threshold, no warning) or triggered if your rule is “<= threshold” (state should match your defined rule).

deductStock(itemID:string):boolean	SM-23	itemID = "C011" (quantity becomes below lowStockThreshold after deduction)	Low-stock warning scenario.	“success” (true). Quantity reduced and low-stock condition flagged (e.g., warning state/log/notification generated as per your implementation).
getCatalogue():List<catalogueItem>	SM-24	(none)	localDatabase contains items after updateLocalStorage().	“success”. Non-null list returned; items reflect localDatabase values.
getCatalogue():List<catalogueItem>	SM-25	(none)	localDatabase empty.	“success”. Empty list returned (not null).
getOrderStatus(orderID:string) : string	OS-1	orderID = "O1001"	Order exists and is currently in a valid non-delivered state (e.g., “PROCESSING” / “IN_TRANSIT”).	“success” (non-empty string returned). Returned status matches the stored status for O1001.
getOrderStatus(orderID:string) : string	OS-2	orderID = "O1002"	Order exists and has been delivered.	“success” (non-empty string returned). Returned status is “DELIVERED” (or the delivered-equivalent defined by the system).

getOrderStatus(orderID:string) : string	OS-3	orderID = "UNKNOWN"	Order does not exist in the system.	Failure (return value indicates “not found”, e.g., “NOT_FOUND”). No system state changes.
getOrderStatus(orderID:string) : string	OS-4	orderID = "" (empty)	Empty orderID sent.	Failure (return value indicates invalid orderID / “NOT_FOUND”). No system state changes.
getOrderStatus(orderID:string) : string	OS-5	orderID = " " (whitespace)	Whitespace-only orderID sent.	Failure (return value indicates invalid orderID / “NOT_FOUND”). No system state changes.
getUndeliveredOrders() : string[]	OS-6	(none)	There are one or more undelivered orders in the system.	“success” (non-null array returned). Array length > 0 and contains only orderIDs whose status is not “DELIVERED”.
getUndeliveredOrders() : string[]	OS-7	(none)	All orders have status “DELIVERED”.	“success” (non-null array returned). Empty array returned (length = 0).

getUndeliveredOrders() : string[]	OS-8	(none)	System has mixed order statuses; validate filtering logic.	“success”. Returned array contains no delivered orders. Every returned orderID, when passed to getOrderStatus(orderID), returns a non-delivered status.
getUndeliveredOrders() : string[]	OS-9	(none)	No orders exist in the system.	“success” (non-null array returned). Empty array returned (length = 0).
getUndeliveredOrders() : string[]	OS-10	(none)	Duplicate prevention. System contains multiple orders; ensure no duplicates returned.	“success”. Returned array contains unique orderIDs only (no duplicates).
getCatalogue() : List<catalogueItem>	INV-1	(none)	Catalogue contains one or more items.	“success” (non-null List returned). List size > 0 and each catalogueItem has valid non-empty id/name fields.
getCatalogue() : List<catalogueItem>	INV-2	(none)	Catalogue empty scenario (no items loaded / after reset).	“success” (empty List returned). List is not null.

getCatalogue() : List<catalogueItem>	INV-3	(none)	Data consistency check for returned items.	“success”. Returned list contains no duplicate item IDs; quantities are non-negative.
deductStock(itemID:string) : boolean	INV-4	itemID = "C001"	Item exists and quantity > 0.	“success” (Boolean returned is “true”). Quantity for C001 reduced by 1.
deductStock(itemID:string) : boolean	INV-5	itemID = "C003"	Item exists but quantity = 0 (out of stock).	Failure (Boolean returned is “false”). Quantity remains 0.
deductStock(itemID:string) : boolean	INV-6	itemID = "UNKNOWN"	Item does not exist in catalogue.	Failure (Boolean returned is “false”). No stock changed.
deductStock(itemID:string) : boolean	INV-7	itemID = "" (empty)	Empty itemID sent.	Failure (Boolean returned is “false”). No stock changed.
deductStock(itemID:string) : boolean	INV-8	itemID = " " (whitespace)	Whitespace-only itemID sent.	Failure (Boolean returned is “false”). No stock changed.
deductStock(itemID:string) : boolean	INV-9	itemID = "C001" (quantity initially 1)	Boundary case: quantity reaches 0 after deduction.	“success” (true). Quantity becomes 0, never negative.

deductStock(itemID:string) : boolean	INV-10	itemID = "C001" (quantity initially 0, then deduct called again)	Prevent negative stock.	Failure (false). Quantity remains 0. No negative values allowed.
SA_SystemSimulation() : constructor	SA-1	(none)	Simulator initialised.	“success”. catalogueSimulator and orderSimulator created and not null. Initial simulated data available for testing.
generateSimulatedData() : void	SA-2	(none)	Generate catalogue and order simulation data.	“success”. Catalogue populated with at least one catalogueItem and at least one simulated order exists.
generateSimulatedData() : void	SA-3	(none)	Method called twice consecutively.	“success”. Duplicate data not created (catalogue items and orders remain unique).
placeRestockOrder(merchantID:string, orderDetails:string) : string	SA-4	merchantID = "M001" orderDetails = “{item:C001, qty:10}"	Valid merchant and order details.	“success”. Unique orderID returned. Order added into orderSimulator with initial status (e.g., PROCESSING).

placeRestockOrder(merchantID:string, orderDetails:string) : string	SA-5	merchantID = "UNKNOWN" orderDetails = "{item:C001, qty:10}"	Merchant does not exist.	Failure (return value indicates merchant not recognised). No order created.
placeRestockOrder(merchantID:string, orderDetails:string) : string	SA-6	merchantID = "M001" orderDetails = ""	Empty order details sent.	Failure (return value indicates invalid order details). No order created.
trackDelivery(orderID:string) : string	SA-7	orderID = "O1001"	Existing order.	“success”. Returns delivery state (e.g., PROCESSING / IN_TRANSIT / DELIVERED).
trackDelivery(orderID:string) : string	SA-8	orderID = "UNKNOWN"	Order does not exist.	Failure (return value indicates NOT_FOUND).
trackDelivery(orderID:string) : string	SA-9	orderID = ""	Empty orderID sent.	Failure (invalid orderID response).
getOrderStatus(orderID:string) : string	SA-10	orderID = "O1001"	Existing order queried through OrderStatus interface.	“success”. Returned status matches simulator stored status.
getOrderStatus(orderID:string) : string	SA-11	orderID = "UNKNOWN"	Unknown order.	Failure (“NOT_FOUND”).
getUndeliveredOrders() : string[]	SA-12	(none)	Simulator contains undelivered orders.	“success”. Returned array contains only orders not marked DELIVERED.

getUndeliveredOrders() : string[]	SA-13	(none)	All orders delivered.	“success”. Empty array returned (not null).
queryOutstandingBalance(merchantID:string) : double	SA-14	merchantID = "M001"	Merchant exists.	“success”. Non-negative numeric balance returned.
queryOutstandingBalance(merchantID:string) : double	SA-15	merchantID = "UNKNOWN"	Merchant does not exist.	Failure behaviour defined (0.0 returned or error indicator).
getInvoice(orderID:string) : string	SA-16	orderID = "O1001"	Existing order invoice requested.	“success”. Non-empty invoice string returned referencing correct orderID.
getInvoice(orderID:string) : string	SA-17	orderID = "UNKNOWN"	Unknown order.	Failure (“NOT_FOUND”).
deductStock(itemID:string) : boolean	SA-18	itemID = "C001"	Valid item with quantity > 0.	“success” (Boolean returned is true). Quantity reduced by 1.
deductStock(itemID:string) : boolean	SA-19	itemID = "C003"	Out-of-stock item.	Failure (false). Quantity remains 0.
deductStock(itemID:string) : boolean	SA-20	itemID = "UNKNOWN"	Unknown item.	Failure (false). No changes made.
getCatalogue() : List<catalogueItem>	SA-21	(none)	Catalogue contains simulated items.	“success”. Non-null list returned with valid catalogueItem entries.
getCatalogue() : List<catalogueItem>	SA-22	(none)	Catalogue empty scenario.	“success”.

				Empty list returned (not null).
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-1	email = customer@test.com content = "Your order has been confirmed." reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Valid email request with all fields present.	“success” (the Boolean returned is “true”). Email is accepted for sending.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-2	email = "invalid-email" content = "Test message" reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Invalid email format sent.	Failure (the Boolean returned is “false”). No email is sent.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-3	email = "" (empty) content = "Test message" reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Empty email address sent.	Failure (the Boolean returned is “false”). No email is sent.

produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-4	email = customer@test.com content = "" (empty) reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Empty email content sent.	Failure (the Boolean returned is “false”). No email is sent.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-5	email = customer@test.com content = "Test message" reference = "" (empty) sender = "IPOS_CA" subsystem = "IPOS_CA"	Empty reference sent.	Failure (the Boolean returned is “false”) or “success” (true) depending on implementation rule; behaviour must be consistent and documented. If rejected, no email is sent.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-6	email = "customer@test.com" content = "Test message" reference = "O1001" sender = "" (empty) subsystem = "IPOS_CA"	Empty sender sent.	Failure (the Boolean returned is “false”). No email is sent.

produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-7	email = customer@test.com content = "Test message" reference = "O1001" sender = "IPOS_CA" subsystem = "" (empty)	Empty subsystem identifier sent.	Failure (the Boolean returned is “false”). No email is sent.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	IE-8	email = customer@test.com content = "A" repeated 5000 times reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Very large email content (boundary test).	“success” (true) if large messages are allowed, otherwise Failure (false) with a “content too long” rule; behaviour must be consistent and documented.
EmailServiceSimulator() : constructor	EMS-1	(none)	Simulator initialised.	“success”. sentLog is created and empty (size = 0).
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	EMS-2	email = "customer@test.com" content = "Order confirmed." reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Valid email request.	“success” (Boolean returned is “true”). Email added to sentLog with correct details.

produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	EMS-3	email = "invalid-email" content = "Test" reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Invalid email format.	Failure (Boolean returned is “false”). sentLog size unchanged.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	EMS-4	email = "customer@test.com" content = "" reference = "O1001" sender = "IPOS_CA" subsystem = "IPOS_CA"	Empty email content.	Failure (Boolean returned is “false”). Email not recorded.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	EMS-5	email = "customer@test.com" content = "Test message" reference = "REF1" sender = "IPOS_CA" subsystem = "IPOS_CA"	Multiple valid emails sent sequentially.	“success”. sentLog size increases correctly and preserves send order.
getEmailWithID(details:string) : string	EMS-6	details = "REF1"	Retrieve existing stored email entry.	“success”. Correct email record returned matching REF1.
getEmailWithID(details:string) : string	EMS-7	details = "UNKNOWN_REF"	Request email not stored in simulator.	Failure. (returns “NOT_FOUND” or empty string depending on implementation rule).

getEmailWithID(details:string) : string	EMS-8	details = ""	Empty reference sent.	Failure. Invalid reference indicator returned.
produceEmail(email:string, content:string, reference:string, sender:string, subsystem:string) : boolean	EMS-9	email = "customer@test.com" content = "Boundary message" reference = "REF-LARGE" sender = "IPOS_CA" subsystem = "IPOS_CA"	Boundary test — very large message content.	Either accepted (true) or rejected (false) depending on system limits; behaviour must remain consistent.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-1	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Valid payment request.	“success”. Return value indicates payment approved and includes transaction reference.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-2	merchantID = "UNKNOWN" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Merchant does not exist.	Failure (return value indicates “MERCHANT_NOT_FOUND”) . No payment processed.

requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-3	merchantID = "M001" orderID = "UNKNOWN" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Order does not exist.	Failure (return value indicates “ORDER_NOT_FOUND”).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-4	merchantID = "M001" orderID = "O1001" fullName = "" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Missing payer name.	Failure (return value indicates invalid customer details).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-5	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Missing address.	Failure (invalid payer details).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-6	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London" cardDetails = null amount = 120.50	Card details missing.	Failure (return value indicates invalid card details).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-7	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London" cardDetails = ["123"] amount = 120.50	Card details incomplete.	Failure (invalid card data).

requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-8	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London" cardDetails = ["4111111111111111","12/28","123"] amount = 0	Zero payment amount.	Failure (return value indicates invalid amount).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-9	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London" cardDetails = ["4111111111111111","12/28","123"] amount = -10	Negative payment amount.	Failure (invalid amount).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-10	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London" cardDetails = ["4111111111111111","12/28","123"] amount = 99999999	Extremely large payment amount (boundary test).	Failure or approval depending on configured payment limit; behaviour must remain consistent.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string[], amount:double) : string	IPP-11	merchantID = "" orderID = "" fullName = "" address = "" cardDetails = null amount = 50	Multiple invalid parameters.	Failure (invalid request). No payment processed.
PaymentSimulator() : constructor	PS-1	(none)	Simulator initialised.	“success”. connectedToBank initialised (true by default if simulator starts online) and transaction log is created and empty.

authoriseTransaction(card:string, amount:double) : boolean	PS-2	card = "4111111111111111" amount = 50.00	Valid card data and positive amount.	“success” (Boolean returned is “true”). Transaction is authorised.
authoriseTransaction(card:string, amount:double) : boolean	PS-3	card = "" amount = 50.00	Empty card string.	Failure (Boolean returned is “false”). Transaction not authorised.
authoriseTransaction(card:string, amount:double) : boolean	PS-4	card = "4111111111111111" amount = 0	Amount is zero (invalid).	Failure (Boolean returned is “false”).
authoriseTransaction(card:string, amount:double) : boolean	PS-5	card = "4111111111111111" amount = -10.00	Negative amount.	Failure (Boolean returned is “false”).
authoriseTransaction(card:string, amount:double) : boolean	PS-6	card = "123" amount = 25.00	Invalid/too short card number.	Failure (Boolean returned is “false”).
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-7	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Valid payment through simulator.	“success”. Return value indicates approval and includes a transaction reference. Transaction recorded in simulator log.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-8	Parameter values: merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = null amount = 120.50	Missing card details.	Failure (return value indicates invalid card details). No transaction recorded.

requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-9	merchantID = "M001" orderID = "O1001" fullName = "" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Missing payer name.	Failure (return value indicates invalid payer details). No transaction recorded.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-10	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 0	Zero amount.	Failure (return indicates invalid amount). No transaction recorded.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-11	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Bank connection unavailable (connectedToBank = false).	Failure (return value indicates “SERVICE_UNAVAILABLE” / “BANK_OFFLINE”). No transaction recorded.
requestPayment(merchantID:string, orderID:string, fullName:string, address:string, cardDetails:string, amount:double) : string	PS-12	merchantID = "M001" orderID = "O1001" fullName = "Talip Tun" address = "London, UK" cardDetails = ["4111111111111111","12/28","123"] amount = 120.50	Duplicate payment attempt for the same orderID (double-charge prevention).	Failure (return value indicates “ALREADY_PAID” / “DUPLICATE_PAYMENT”) OR “success” but generates a new transaction based on your rule; behaviour must be consistent and documented.

placeRestockOrder(merchantID:string, orderDetails:string) : string	OR-1	merchantID = "M001" orderDetails = "{item:C001,qty:10}"	Valid merchant and valid order details.	“success” (non-empty orderID string returned). New order created with correct merchantID and details.
placeRestockOrder(merchantID:string, orderDetails:string) : string	OR-2	merchantID = "UNKNOWN" orderDetails = "{item:C001,qty:10}"	Merchant does not exist.	Failure (return value indicates “MERCHANT_NOT_FOUND” / error string). No order created.
placeRestockOrder(merchantID:string, orderDetails:string) : string	OR-3	merchantID = "M001" orderDetails = "" (empty)	Empty order details sent.	Failure (return value indicates invalid order details). No order created.
placeRestockOrder(merchantID:string, orderDetails:string) : string	OR-4	merchantID = "" (empty) orderDetails = "{item:C001,qty:10}"	Empty merchantID sent.	Failure (return value indicates invalid merchantID). No order created.
placeRestockOrder(merchantID:string, orderDetails:string) : string	OR-5	merchantID = "M001" orderDetails = "{item:C001,qty:-5}"	Invalid quantity in order details (negative).	Failure (return value indicates invalid order details). No order created.
trackDelivery(orderID:string) : string	OR-6	orderID = "O1001"	Existing order; delivery tracking requested.	“success” (non-empty status string returned). Status is a valid delivery state (e.g., PROCESSING / IN_TRANSIT / DELIVERED).

trackDelivery(orderID:string) : string	OR-7	orderID = "UNKNOWN"	Order does not exist.	Failure (return value indicates “NOT_FOUND”). No state changes.
trackDelivery(orderID:string) : string	OR-8	orderID = "" (empty)	Empty orderID sent.	Failure (return value indicates invalid orderID / “NOT_FOUND”).
queryOutstandingBalance(merchantID:string) : double	OR-9	merchantID = "M001"	Merchant exists; outstanding balance requested.	“success” (numeric value returned). Balance is non-negative.
queryOutstandingBalance(merchantID:string) : double	OR-10	merchantID = "UNKNOWN"	Merchant does not exist.	Failure behaviour defined (0.0 returned or error sentinel). No state changes.
queryOutstandingBalance(merchantID:string) : double	OR-11	merchantID = "" (empty)	Empty merchantID sent.	Failure behaviour defined (0.0 returned or error sentinel). No state changes.
getInvoice(orderID:string) : string	OR-12	orderID = "O1001"	Invoice requested for an existing order.	“success” (non-empty string returned). Invoice contains correct order reference.
getInvoice(orderID:string) : string	OR-13	orderID = "UNKNOWN"	Invoice requested for unknown order.	Failure (return value indicates “NOT_FOUND” / empty invoice). No state changes.

getInvoice(orderID:string) : string	OR-14	orderID = "" (empty)	Empty orderID sent.	Failure (return value indicates invalid orderID / “NOT_FOUND”).
getAccStatus(merchantID:string, status:string) : boolean	OR-15	merchantID = "M001" status = "ACTIVE"	Merchant account is active.	“success” (Boolean returned is “true”).
getAccStatus(merchantID:string, status:string) : boolean	OR-16	merchantID = "M001" status = "SUSPENDED"	Merchant account is not active.	Failure (Boolean returned is “false”).
getAccStatus(merchantID:string, status:string) : boolean	OR-17	merchantID = "UNKNOWN" status = "ACTIVE"	Merchant does not exist.	Failure (Boolean returned is “false”).
getAccStatus(merchantID:string, status:string) : boolean	OR-18	merchantID = "M001" status = "" (empty)	Empty status sent.	Failure (Boolean returned is “false”).
viewDiscountPlan(merchantID:string, plan:DiscountPlan) : boolean	OR-19	merchantID = "M001" plan = DiscountPlan("STANDARD", 0.05)	Merchant exists and discount plan provided.	“success” (Boolean returned is “true”). Discount plan can be viewed/returned as per implementation rule.
viewDiscountPlan(merchantID:string, plan:DiscountPlan) : boolean	OR-20	merchantID = "M001" plan = null	Missing DiscountPlan object.	Failure (Boolean returned is “false”).
viewDiscountPlan(merchantID:string, plan:DiscountPlan) : boolean	OR-21	merchantID = "UNKNOWN" plan = DiscountPlan("STANDARD", 0.05)	Merchant does not exist.	Failure (Boolean returned is “false”).

viewCreditLimit(merchantID:string) : boolean	OR-22	merchantID = "M001"	Merchant exists and has a defined credit limit.	“success” (Boolean returned is “true”). Credit limit is available for viewing (as defined by implementation).
viewCreditLimit(merchantID:string) : boolean	OR-23	merchantID = "UNKNOWN"	Merchant does not exist.	Failure (Boolean returned is “false”). No state changes.
viewCreditLimit(merchantID:string) : boolean	OR-24	merchantID = "" (empty)	Empty merchantID sent.	Failure (Boolean returned is “false”). No state changes.