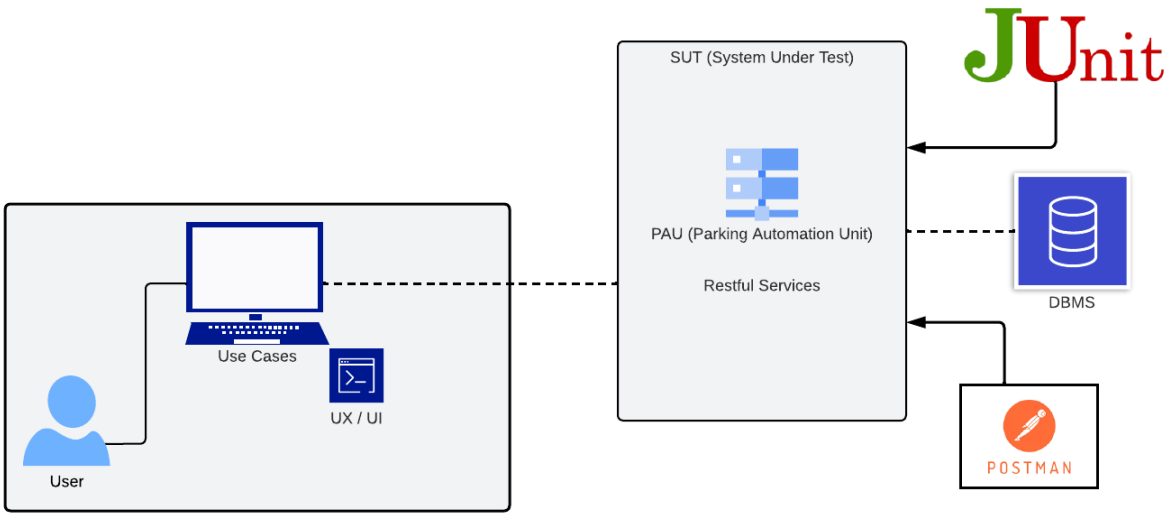
Ahmet Efe Kutbay - 200503019
Altay Yavuz Elmas - 200503046

Systemtestdokument

Chapter 1: Systemüberblick



Chapter 2: Systemtestfälle

ParkingLotController Test Cases

a) createParkingLot

System Test Case ID Name	S_TC_PAU_REQ_1
Preconditions	Web Service ist gegeben: 1. Request: POST createParkingLot(ParkingLotCreateRequest request) 2. Response: ParkingLotDto ○ Data: “ParkingLotDto” ■ Integer lotID ■ String name ■ AddressDto addressDto ■ List<ParkSlotDto> parkSlotDtoList ○ Data “AddressDto” ■ Integer addressID ■ String street

	■ String city ■ String postCode ■ String country ■ Integer lotID ○ Data “ParkSlotDto” ■ Integer slotID ■ boolean isAvailable ■ Integer lotID ○ Data: “ParkingLotCreateRequest” ■ String name ■ String street ■ String city ■ String postCode ■ String country ■ Integer numberOfSlots 3. request hat folgende Attributen: ○ name = “TestName” ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ○ numberOfSlots = 50 4. App port number: 8080 (default) 5. System is up and running.
Test Steps	1. POST createParkingLot(ParkingLotCreateRequest request) ist mit request aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	none
Expected Result	ParkingLotDto: ● lotID = <generated> ● name = “TestName” ● addressDto: ○ addressID = <generated> ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated>

	○ lotID = <generated> ○ available = true
Actual Result	ParkingLotDto: ● lotID = <generated> ● name = “TestName” ● addressDto: ○ addressID = <generated> ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = <generated> ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_1 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_1
Preconditions	● Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotCreateRequest wird hergestellt. 3. Junit Test Case ruft createParkingLot mit request auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	request hat folgende Attributen: ● name = “TestParkingLot” ● street = “TestStreet” ● city = “TestCity” ● postCode = “TestPostCode” ● country = “TestCountry” ● numberOfSlots = 50
Expected Result	ParkingLotDto: ● lotID = <generated> ● name = “TestName” ● addressDto: ○ addressID = <generated> ○ street = “TestStreet”

	○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = <generated> ○ available = true
Actual Result	ParkingLotDto: ● lotID = <generated> ● name = “TestName” ● addressDto: ○ addressID = <generated> ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = <generated> ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_1 (Siehe Architekturdokumentation)

b) getAll

System Test Case ID Name	S_TC_PAU_REQ_2
Preconditions	1. Web Service ist gegeben: 2. Request: GET getAll() 3. Response: List<ParkingLotDto> ○ Data: “ParkingLotDto” ■ Integer lotID ■ String name ■ AddressDto addressDto ■ List<ParkSlotDto> parkSlotDtoList ○ Data “AddressDto” ■ Integer addressID ■ String street ■ String city ■ String postCode ■ String country ■ Integer lotID ○ Data “ParkSlotDto” ■ Integer slotID

	■ boolean isAvailable ■ Integer lotID 4. ParkingLot1 ist konfiguriert: ○ lotID = <generated> ○ name = “TestName” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = ParkingLot1 ○ parkSlotList mit 50 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot1 ■ ticket = NULL 5. ParkingLot2 ist konfiguriert: ○ lotID = <generated> ○ name = “TestName2” ○ address: ■ addressID = <generated> ■ street = “TestStreet2” ■ city = “TestCity2” ■ postCode = “TestPostCode2” ■ country = “TestCountry2” ■ parkingLot = ParkingLot2 ○ parkSlotList mit 50 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot2 ■ ticket = NULL 6. App port number: 8080 (default) 7. System is up and running.
Test Steps	1. GET getAll() ist angerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	none
Expected Result	● ParkingLotDto1 ○ lotID = <generated>

	○ name = “TestName2” ○ addressDto: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = <generated> ■ available = true ● ParkingLotDto2: ○ lotID = <generated> ○ name = “TestName2” ○ addressDto: ■ addressID = <generated> ■ street = “TestStreet2” ■ city = “TestCity2” ■ postCode = “TestPostCode2” ■ country = “TestCountry2” ○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = <generated> ■ available = true
Actual Result	● ParkingLotDto1: ○ lotID = <generated> ○ name = “TestName” ○ addressDto: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = <generated> ■ available = true ● ParkingLotDto2: ○ lotID = <generated> ○ name = “TestName2” ○ addressDto: ■ addressID = <generated> ■ street = “TestStreet2” ■ city = “TestCity2” ■ postCode = “TestPostCode2” ■ country = “TestCountry2”

	○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = <generated> ■ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_2

Component Integration Test Case ID Name	CI_TC_PAU_REQ_2
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat 5 Zeilen. ● ParkingLot1 bis ParkingLot5 ist konfiguriert: ○ lotID = 1 bis 5 ○ name = “TestName” ○ addressDto: ■ addressID = 1 ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = 1 bis 5 ■ available = true
Test Steps	● Es wird sichergestellt, dass parkingLotService gemockt ist. ● Ein ParkingLotDtoList wird hergestellt. ● Junit Test Case ruft getAll auf. ● Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	● ParkingLot1 bis ParkingLot5 ParkingLotDto1 bis 5: ● lotID = 0 bis 4 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry”

	● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 0 bis 4 ○ available = true
Expected Result	List<ParkingLotDto>: ParkingLotDto1 bis 5 ● lotID = 0 bis 4 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 0 bis 4 ○ available = true
Actual Result	List<ParkingLotDto>: ParkingLotDto1 bis 5 ● lotID = 0 bis 4 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 0 bis 4 ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_2 (Siehe Architekturdokumentation)

c) getParkingLotById

System Test Case ID Name	S_TC_PAU_REQ_3
Preconditions	Web Service ist gegeben: 1. Request: GET getParkingLotByID (Integer lotID) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse”

	○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. ParkingLot1 ist konfiguriert: ● lotID = <generated> ● name = “TestName” ● addressDto: ○ addressID = <generated> ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = <generated> ○ available = true 4. App port number: 8080 (default) 5. System is up and running.
Test Steps	1. GET getParkingLotByID(Integer lotID) ist mit lotID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse:

	• parkingLotDto = ParkingLotDto
	• message = “200 OK”
	• success = true
	
	ParkingLotDto:
	• lotID = <generated>
	• name = “TestName”
	• addressDto:
	  ○ addressID = <generated>
	  ○ street = “TestStreet”
	  ○ city = “TestCity”
	  ○ postCode = “TestPostCode”
	  ○ country = “TestCountry”
	
	• parkSlotDtoList mit 50 ParkSlotDtos:
	  ○ slotID = <generated>
	  ○ lotID = <generated>
	  ○ available = true

Actual Result	ParkingLotResponse:
	• parkingLotDto = ParkingLotDto
	• message = “200 OK”
	• success = true
	
	ParkingLotDto:
	• lotID = <generated>
	• name = “TestName”
	• addressDto:
	  ○ addressID = <generated>
	  ○ street = “TestStreet”
	  ○ city = “TestCity”
	  ○ postCode = “TestPostCode”
	  ○ country = “TestCountry”
	
	• parkSlotDtoList mit 50 ParkSlotDtos:
	  ○ slotID = <generated>
	  ○ lotID = <generated>
	  ○ available = true

Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_3 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_3

Preconditions	• Web Service Anruf ist gemockt.
	• Database Table “parking_lot” hat eine Zeile.
	• ParkingLot1:
	  ○ lotID = 1
	  ○ name = “TestName”

	○ addressDto: ■ addressID = 1 ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ○ parkSlotDtoList mit 50 ParkSlotDtos: ■ slotID = <generated> ■ lotID = 1 ■ available = true
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotDto wird hergestellt. 3. Ein ParkingLotResponse wird hergestellt. 4. Junit Test Case ruft getParkingLotById mit lotID = 1 auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ○ available = true
Expected Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestName” ● addressDto:

	○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ○ available = true
Actual Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_3 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_3_E1
Preconditions	Web Service ist gegeben: 1. Request: GET getParkingLotByID (Integer lotID) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto”

	○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. Parking Lot mit lotID = 1 existiert nicht. 4. App port number: 8080 (default) 5. System is up and running.
Test Steps	1. GET getParkingLotByID(Integer lotID) ist mit lotID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_3 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_3_E1
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat keine Zeile mit lotID = 1.

Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotResponse wird hergestellt. 3. Junit Test Case ruft getParkingLotById mit lotID = 1 auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_3 (Siehe Architekturdokumentation)

d) deleteParkingLot

System Test Case ID Name	S_TC_PAU_REQ_4
Preconditions	Web Service ist gegeben: 1. Request: DELETE deleteParkingLot (Integer lotID) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList

	● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. Parking Lot mit lotID = 7 existiert. 4. ParkingLot7 ist konfiguriert: ● lotID = <generated> ● name = “TestName2” ● address: ○ addressID = <generated> ○ street = “TestStreet2” ○ city = “TestCity2” ○ postCode = “TestPostCode2” ○ country = “TestCountry2” ○ parkingLot = ParkingLot2 ● parkSlotList mit 50 ParkSlots: ○ slotID = <generated> ○ isAvailable = true ○ parkingLot = ParkingLot7 ○ ticket = NULL 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. DELETE deleteParkingLot(Integer lotID) ist mit lotID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 7 ● name = “TestName2” ● addressDto:

	○ addressID = 7 ○ street = “TestStreet2” ○ city = “TestCity2” ○ postCode = “TestPostCode2” ○ country = “TestCountry2” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 7 ○ available = true
Actual Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 7 ● name = “TestName2” ● addressDto: ○ addressID = 7 ○ street = “TestStreet2” ○ city = “TestCity2” ○ postCode = “TestPostCode2” ○ country = “TestCountry2” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 7 ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_4
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat eine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotResponse wird hergestellt. 3. Ein ParkingLotDto wird hergestellt. 4. Junit Test Case ruft deleteParkingLot mit lotID = 1

	auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ○ available = true
Expected Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ○ available = true
Actual Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto:

	● lotID = 1 ● name = “TestName” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 1 ● available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_4_E1
Preconditions	Web Service ist gegeben: 1. Request: DELETE deleteParkingLot (Integer lotID) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID

	3. Parking Lot mit lotID = 1 existiert nicht. 4. App port number: 8080 (default) 5. System is up and running.
Test Steps	1. DELETE deleteParkingLot(Integer lotID) ist mit lotID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_4_E1
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat keine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotResponse wird hergestellt. 3. Junit Test Case ruft deleteParkingLot mit lotID = 1 auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	ParkingLotResponse:

	• parkingLotDto = null • message = “Message” • success = false
Expected Result	ParkingLotResponse: • parkingLotDto = null • message = “Message” • success = false
Actual Result	ParkingLotResponse: • parkingLotDto = null • message = “Message” • success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_4_E2
Preconditions	Web Service ist gegeben: 1. Request: DELETE deleteParkingLot (Integer lotID) 2. Response: ParkingLotResponse ○ Data: “ParkingLotResponse” ■ ParkingLotDto parkingLotDto ■ boolean isSuccess ■ String message ○ Data: “ParkingLotDto” ■ Integer lotID ■ String name ■ AddressDto addressDto ■ List<ParkSlotDto> parkSlotDtoList ○ Data “AddressDto” ■ Integer addressID ■ String street ■ String city ■ String postCode ■ String country ■ Integer lotID ○ Data “ParkSlotDto” ■ Integer slotID ■ boolean isAvailable ■ Integer lotID 3. Parking Lot mit lotID = 6 existiert und hat ein Ticket 4. ParkingLot6 ist konfiguriert: ○ lotID = <generated> ○ name = “TestName”

	○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = ParkingLot6 ○ parkSlotList mit 49 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot6 ■ ticket = NULL ○ und ein ParkSlot mit: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot6 ■ ticket: ● ticketID: <generated> ● entryDate: <generated> ● exitDate: <generated> ● isValid: false ● vehicle: ○ licencePlate: “35ABD123” ○ vehicleType: “AUTO” ○ ticketList hat nur ein Element ticket ● parkSlot: ○ slotID: <generated> ○ isAvailable: true ○ parkingLot: ParkingLot6 ○ ticket: ticket 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. DELETE deleteParkingLot(Integer lotID) ist mit lotID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “This Parking Lot has tickets” ● success = false

Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “This Parking Lot has tickets” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_4_E2
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat eine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotResponse wird hergestellt. 3. Junit Test Case ruft deleteParkingLot mit lotID = 1 auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Message” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_4 (Siehe Architekturdokumentation)

e) updateParkingLot

System Test Case ID Name	S_TC_PAU_REQ_5
Preconditions	Web Service ist gegeben: 1. Request: PATCH updateParkingLot (Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. ParkingLotUpdateRequest ist konfiguriert: ○ name = “NewName” ○ street = “NewStreet” ○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = ”NewCountry” ○ numberOfSlots = 50 4. Parking Lot mit lotID = 6 existiert. 5. ParkingLot6 ist konfiguriert: ○ lotID = <generated> ○ name = “TestName” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry”

	■ parkingLot = ParkingLot6 ○ parkSlotList mit 49 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot6 ■ ticket = NULL ○ und ein ParkSlot mit: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = ParkingLot6 ■ ticket: ● ticketID: <generated> ● entryDate: <generated> ● exitDate: <generated> ● isValid: false ● vehicle: ○ licencePlate: “35ABD123” ○ vehicleType: “AUTO” ○ ticketList hat nur ein Element ticket ● parkSlot: ○ slotID: <generated> ○ isAvailable: true ○ parkingLot: ParkingLot6 ○ ticket: ticket 6. App port number: 8080 (default) 7. System is up and running.
Test Steps	1. PATCH updateParkingLot(Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) ist mit lotID und lotUpdateRequest aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 6 ● name = “NewName” ● addressDto: ○ addressID = <generated> ○ street = “NewStreet”

	○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = “NewCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 6 ○ available = true
Actual Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 6 ● name = “NewName” ● addressDto: ○ addressID = <generated> ○ street = “NewStreet” ○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = “NewCountry” ● parkSlotDtoList mit 50 ParkSlotDtos: ○ slotID = <generated> ○ lotID = 6 ○ available = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_5
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat eine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotUpdateRequest wird hergestellt. 3. Ein ParkingLotDto wird hergestellt. 4. Ein ParkingLotResponse wird hergestellt 5. Junit Test Case ruft updateParking mit lotID = 1 und parkingLotUpdateRequest auf. 6. Response Data werden analysiert und die

	erwarteten Daten werden ausgewählt
Post-Condition	none
Test Data	ParkingLotDto: ● lotID = 1 ● name = “TestParkingLot” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ○ lotID = 1 ● List<ParkSlotDto> mit 50 ParkSlotDtos: ○ slotID 0 bis 49 ○ isAvailable = true ○ parkingLot = parkingLot ParkingLotUpdateRequest: ● name = “TestParkingLot2” ● street = “TestStreet2” ● city = “TestCity2” ● postCode = “TestPostCode2” ● country = “TestCountry2” ● numberOfSlots = 50 ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● isSuccess = true ● message = “200 OK”
Expected Result	ParkingLotResponse: ● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestParkingLot” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ○ lotID = 1 ● List<ParkSlotDto> mit 50 ParkSlotDtos: ○ slotID 0 bis 49 ○ isAvailable = true ○ parkingLot = parkingLot
Actual Result	ParkingLotResponse:

	● parkingLotDto = ParkingLotDto ● message = “200 OK” ● success = true ParkingLotDto: ● lotID = 1 ● name = “TestParkingLot” ● addressDto: ○ addressID = 1 ○ street = “TestStreet” ○ city = “TestCity” ○ postCode = “TestPostCode” ○ country = “TestCountry” ○ lotID = 1 ● List<ParkSlotDto> mit 50 ParkSlotDtos: ○ slotID 0 bis 49 ○ isAvailable = true ○ parkingLot = parkingLot
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_5_E1
Preconditions	Web Service ist gegeben: 1. Request: PATCH updateParkingLot (Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto ○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto”

	○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. ParkingLotUpdateRequest ist konfiguriert: ○ name = “NewName” ○ street = “NewStreet” ○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = ”NewCountry” ○ numberOfSlots = 50 4. Parking Lot mit lotID = 1 existiert nicht. 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. PATCH updateParkingLot(Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) ist mit lotID und lotUpdateRequest aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Parking Lot not found” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_5_E1
Preconditions	● Web Service Anruf ist gemockt.

	● Database Table “parking_lot” hat keine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotUpdateRequest wird hergestellt. 3. Ein ParkingLotResponse wird hergestellt 4. Junit Test Case ruft updateParking mit lotID = 1 und parkingLotUpdateRequest auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	ParkingLotUpdateRequest: ● name = “TestParkingLot2” ● street = “TestStreet2” ● city = “TestCity2” ● postCode = “TestPostCode2” ● country = “TestCountry2” ● numberOfSlots = 50 ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_5_E2
Preconditions	Web Service ist gegeben: 1. Request: PATCH updateParkingLot (Integer lotID) 2. Response: ParkingLotResponse ● Data: “ParkingLotResponse” ○ ParkingLotDto parkingLotDto

	○ boolean isSuccess ○ String message ● Data: “ParkingLotDto” ○ Integer lotID ○ String name ○ AddressDto addressDto ○ List<ParkSlotDto> parkSlotDtoList ● Data “AddressDto” ○ Integer addressID ○ String street ○ String city ○ String postCode ○ String country ○ Integer lotID ● Data “ParkSlotDto” ○ Integer slotID ○ boolean isAvailable ○ Integer lotID 3. ParkingLotUpdateRequest ist konfiguriert: ○ name = “NewName” ○ street = “NewStreet” ○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = ”NewCountry” ○ numberOfSlots = 10 4. Parking Lot mit lotID = 8 existiert. 5. TestParkingLot mit lotID = 8 ist konfiguriert: ○ lotID = 8 ○ name = “TestParkingLot” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = TestParkingLot ○ parkSlotList mit 14 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = TestParkingLot ■ ticket = NULL ○ und ein ParkSlot mit: ■ slotID = <generated> ■ isAvailable = false ■ parkingLot = TestParkingLot 6. App port number: 8080 (default) 7. System is up and running.

Test Steps	1. PATCH updateParkingLot(Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) ist mit lotID und lotUpdateRequest aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Park Slots are not empty” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “Park Slots are not empty” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_5_E2
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat eine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotUpdateRequest wird hergestellt. 3. Ein ParkingLotResponse wird hergestellt 4. Junit Test Case ruft updateParking mit lotID = 1 und parkingLotUpdateRequest auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	ParkingLotUpdateRequest: ● name = “TestParkingLot2”

	• street = “TestStreet2” • city = “TestCity2” • postCode = “TestPostCode2” • country = “TestCountry2” • numberOfSlots = 50 ParkingLotResponse: • parkingLotDto = null • isSuccess = false • message = “Message”
Expected Result	ParkingLotResponse: • parkingLotDto = null • isSuccess = false • message = “Message”
Actual Result	ParkingLotResponse: • parkingLotDto = null • isSuccess = false • message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_5_E3
Preconditions	Web Service ist gegeben: 1. Request: PATCH updateParkingLot (Integer lotID) 2. Response: ParkingLotResponse ○ Data: “ParkingLotResponse” ■ ParkingLotDto parkingLotDto ■ boolean isSuccess ■ String message ○ Data: “ParkingLotDto” ■ Integer lotID ■ String name ■ AddressDto addressDto ■ List<ParkSlotDto> parkSlotDtoList ○ Data “AddressDto” ■ Integer addressID ■ String street ■ String city ■ String postCode ■ String country ■ Integer lotID

	○ Data “ParkSlotDto” ■ Integer slotID ■ boolean isAvailable ■ Integer lotID 3. ParkingLotUpdateRequest ist konfiguriert: ○ name = “NewName” ○ street = “NewStreet” ○ city = “NewCity” ○ postCode = “NewPostCode” ○ country = ”NewCountry” ○ numberOfSlots = 50 4. Parking Lot mit lotID = 8 existiert. 5. TestParkingLot mit lotID = 8 ist konfiguriert: ○ lotID = 8 ○ name = “TestParkingLot” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = TestParkingLot i. ○ parkSlotList mit 15 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = TestParkingLot ■ ticket = NULL 6. TestParkingLot hat ein Ticket mit folgenden Attributen: ○ ticketID = 13 ○ entryDate = <generated> ○ exitDate = <generated> ○ isValid = false ○ Vehicle: ■ licencePlate = “55ABC555” ■ vehicleType = “AUTO” ■ ticketList hat ein Ticket mit ticketID = 13 ○ parkSlot = <generated> ○ parkingLot = TestParkingLot 7. App port number: 8080 (default) 8. System is up and running.
Test Steps	1. PATCH updateParkingLot(Integer lotID, ParkingLotUpdateRequest lotUpdateRequest) ist mit lotID und lotUpdateRequest aufgerufen.

	2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● message = “There are tickets associated with this slots.” ● success = false
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● message = “There are tickets associated with this slots.” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_5_E3
Preconditions	● Web Service Anruf ist gemockt. ● Database Table “parking_lot” hat eine Zeile mit lotID = 1.
Test Steps	1. Es wird sichergestellt, dass parkingLotService gemockt ist. 2. Ein ParkingLotUpdateRequest wird hergestellt. 3. Ein ParkingLotResponse wird hergestellt 4. Junit Test Case ruft updateParking mit lotID = 1 und parkingLotUpdateRequest auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	ParkingLotUpdateRequest: ● name = “TestParkingLot2” ● street = “TestStreet2” ● city = “TestCity2” ● postCode = “TestPostCode2” ● country = “TestCountry2” ● numberOfSlots = 50

	ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Expected Result	ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Actual Result	ParkingLotResponse: ● parkingLotDto = null ● isSuccess = false ● message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_5 (Siehe Architekturdokumentation)

TicketController Test Cases

a) **createTicket**

System Test Case ID Name	S_TC_PAU_REQ_6
Preconditions	Web Service ist gegeben: 1. Request: POST createTicket (TicketCreateRequest request) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. TicketCreateRequest ist konfiguriert: ○ licencePlate = “12TEST123” ○ vehicleType = “AUTO”

	○ lotID = 8 ○ slotID = 215 4. Parking Lot mit lotID = 8 existiert. 5. TestParkingLot mit lotID = 8 ist konfiguriert: ○ lotID = 8 ○ name = “TestParkingLot” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = TestParkingLot ○ parkSlotList mit 15 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = TestParkingLot ■ ticket = NULL 6. App port number: 8080 (default) 7. System is up and running.
Test Steps	1. POST createTicket(TicketCreateRequest request) ist request aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = ticketDto ● message = “200 OK” ● success = true TicketDto: ● ticketID = <generated> ● entryDate = <generated> ● exitDate = null ● licencePlate = “12TEST123” ● slotID = 215 ● lotID = 8 ● valid = true
Actual Result	TicketResponse: ● ticketDto = ticketDto

	• message = “200 OK” • success = true TicketDto: • ticketID = <generated> • entryDate = <generated> • exitDate = null • licencePlate = “12TEST123” • slotID = 215 • lotID = 8 • valid = true
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_6
Preconditions	• Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketCreateRequest wird hergestellt. 3. Ein TicketDto wird hergestellt 4. Ein TicketResponse wird hergestellt. 5. Junit Test Case ruft createTicket mit TicketCreateRequest auf. 6. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketCreateRequest: • licencePlate = “ABC1” • vehicleType = “AUTO” • lotID = 5 • slotID = 55 TicketDto: • ticketID = 1 • entryDate = <generated> • exitDate = <generated> • isValid = true • licencePlate = “ABC1” • slotID = 55 • lotID = 5

	TicketResponse: ● ticketDto = TicketDto ● isSuccess = true ● message = “200 OK”
Expected Result	TicketResponse: ● ticketDto = TicketDto ● isSuccess = true ● message = “200 OK”
Actual Result	TicketResponse: ● ticketDto = TicketDto ● isSuccess = true ● message = “200 OK”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_6_E1
Preconditions	Web Service ist gegeben: 1. Request: POST createTicket (TicketCreateRequest request) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. TicketCreateRequest ist konfiguriert: ○ licencePlate = “123TEST123” ○ vehicleType = “AUTO” ○ lotID = 1 ○ slotID = 215 4. Parking Lot mit lotID = 1 existiert nicht. 5. App port number: 8080 (default) 6. System is up and running.

Test Steps	1. POST createTicket(TicketCreateRequest request) ist request aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = null ● message = “Parking Lot not found” ● success = false
Actual Result	TicketResponse: ● ticketDto = null ● message = “Parking Lot not found” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_6_E1
Preconditions	● Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketCreateRequest wird hergestellt. 3. Ein TicketResponse wird hergestellt. 4. Junit Test Case ruft createTicket mit TicketCreateRequest auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketCreateRequest: ● licencePlate = “ABC1” ● vehicleType = “AUTO” ● lotID = 5

	• slotID = 55 TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Expected Result	TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Actual Result	TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_6_E2
Preconditions	Web Service ist gegeben: 1. Request: POST createTicket (TicketCreateRequest request) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. TicketCreateRequest ist konfiguriert: ○ licencePlate = “34TEST34” ○ vehicleType = “AUTO” ○ lotID = 8 ○ slotID = 215 4. Parking Lot mit lotID = 8 existiert. 5. TestParkingLot mit lotID = 8 ist konfiguriert: ○ lotID = 8

	○ name = “TestParkingLot” ○ address: ■ addressID = <generated> ■ street = “TestStreet” ■ city = “TestCity” ■ postCode = “TestPostCode” ■ country = “TestCountry” ■ parkingLot = TestParkingLot ○ parkSlotList mit 14 ParkSlots: ■ slotID = <generated> ■ isAvailable = true ■ parkingLot = TestParkingLot ■ ticket = NULL ○ und mit einem ParkSlot ■ slotID = <generated> ■ isAvailable = false ■ parkingLot = TestParkingLot 6. App port number: 8080 (default) 7. System is up and running.
Test Steps	1. POST createTicket(TicketCreateRequest request) ist request aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = null ● message = “Park Slot not available” ● success = false
Actual Result	TicketResponse: ● ticketDto = null ● message = “Parking Slot not available” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_6_E2
Preconditions	• Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketCreateRequest wird hergestellt. 3. Ein TicketResponse wird hergestellt. 4. Junit Test Case ruft createTicket mit TicketCreateRequest auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketCreateRequest: • licencePlate = “ABC1” • vehicleType = “AUTO” • lotID = 5 • slotID = 55 TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Expected Result	TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Actual Result	TicketResponse: • ticketDto = null • isSuccess = false • message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_6 (Siehe Architekturdokumentation)

b) parkOut

System Test Case ID Name	S_TC_PAU_REQ_7
Preconditions	Web Service ist gegeben: 1. Request: PATCH parkOut (Integer ticketID) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. Ticket mit ticketID = 14 existiert. 4. Ticket mit ticketID = 14 ist konfiguriert: ○ ticketID = 14 ○ entryDate = <generated> ○ exitDate = NULL ○ isValid = true ○ Vehicle: ■ licencePlate = “12TEST123” ■ vehicleType = “AUTO” ■ TicketList hat nur ein Ticket mit ticketID = 14 ○ parkSlot = 215 ○ parkingLot = TestParkingLot 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. PATCH parkOut(Integer ticketID) ist mit ticketID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = ticketDto

	• message = “200 OK” • success = true TicketDto: • ticketID = 14 • entryDate = <generated> • exitDate = <generated> • licencePlate = “12TEST123” • slotID = 215 • lotID = 8 • valid = false
Actual Result	TicketResponse: • ticketDto = ticketDto • message = “200 OK” • success = true TicketDto: • ticketID = 14 • entryDate = <generated> • exitDate = <generated> • licencePlate = “12TEST123” • slotID = 215 • lotID = 8 • valid = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_7
Preconditions	• Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketDto wird hergestellt 3. Ein TicketResponse wird hergestellt. 4. Junit Test Case ruft parkOut mit ticketID auf. 5. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketDto: • ticketID = 1 • entryDate = <generated>

	• exitDate = <generated> • isValid = true • licencePlate = “ABC1” • slotID = 55 • lotID = 5 TicketResponse: • ticketDto = TicketDto • isSuccess = true • message = “200 OK”
Expected Result	TicketResponse: • ticketDto = TicketDto • isSuccess = true • message = “200 OK”
Actual Result	TicketResponse: • ticketDto = TicketDto • isSuccess = true • message = “200 OK”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_7_E1
Preconditions	Web Service ist gegeben: 1. Request: PATCH parkOut (Integer ticketID) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. Ticket mit ticketID = 20 existiert nicht. 4. App port number: 8080 (default) 5. System is up and running.

Test Steps	1. PATCH parkOut(Integer ticketID) ist mit ticketID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = null ● message = “Ticket not found” ● success = false
Actual Result	TicketResponse: ● ticketDto = null ● message = “200 OK” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_7_E1
Preconditions	● Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketResponse wird hergestellt. 3. Junit Test Case ruft parkOut mit ticketID auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”
Expected Result	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”

Actual Result	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

System Test Case ID Name	S_TC_PAU_REQ_7_E2
Preconditions	Web Service ist gegeben: 1. Request: PATCH parkOut (Integer ticketID) 2. Response: TicketResponse ○ Data: “TicketResponse” ■ TicketDto ticketDto ■ boolean isSuccess ■ String message ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. Ticket mit ticketID = 14 existiert. 4. Ticket mit ticketID = 14 ist konfiguriert: ○ ticketID = 14 ○ entryDate = <generated> ○ exitDate = <generated> ○ isValid = false ○ Vehicle: ■ licencePlate = “12TEST123” ■ vehicleType = “AUTO” ■ TicketList hat nur ein Ticket mit ticketID = 14 ○ parkSlot = 215 ○ parkingLot = TestParkingLot 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. PATCH parkOut(Integer ticketID) ist mit ticketID aufgerufen. 2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.

Post-Condition	none
Test Data	none
Expected Result	TicketResponse: ● ticketDto = null ● message = “This ticket is not valid.” ● success = false
Actual Result	TicketResponse: ● ticketDto = ticketDto ● message = “This ticket is not valid.” ● success = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_7_E2
Preconditions	● Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass ticketService gemockt ist. 2. Ein TicketResponse wird hergestellt. 3. Junit Test Case ruft parkOut mit ticketID auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”
Expected Result	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”
Actual Result	TicketResponse: ● ticketDto = null ● isSuccess = false ● message = “Message”
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_7 (Siehe Architekturdokumentation)

c) getAll

System Test Case ID Name	S_TC_PAU_REQ_8
Preconditions	Web Service ist gegeben: 1. Request: GET getAll() 2. Response: List<TicketDto> ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. Ticket mit ticketID = 14 ist konfiguriert: ○ ticketID = 14 ○ entryDate = <generated> ○ exitDate =<generated> ○ isValid = false ○ Vehicle: ■ licencePlate = “12TEST123” ■ vehicleType = “AUTO” ■ TicketList hat nur ein Ticket mit ticketID = 14 ○ parkSlot = 215 ○ parkingLot = TestParkingLot 4. Ticket mit ticketID = 13 ist konfiguriert: ○ ticketID = 13 ○ entryDate = <generated> ○ exitDate =<generated> ○ isValid = false ○ Vehicle: ■ licencePlate = “55ABC555” ■ vehicleType = “AUTO” ■ TicketList hat nur ein Ticket mit ticketID = 13 ○ parkSlot = 215 ○ parkingLot = TestParkingLot 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. PATCH parkOut(Integer ticketID) ist mit ticketID aufgerufen. 2. Response Data werden analysiert und die

	erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	List<TicketDto>: ● TicketDto1: ○ ticketID = 14 ○ entryDate = <generated> ○ exitDate = <generated> ○ licencePlate = “12TEST123” ○ slotID = 215 ○ lotID = 8 ○ valid = false ● TicketDto2: ○ ticketID = 13 ○ entryDate = <generated> ○ exitDate = <generated> ○ licencePlate = “55ABC555” ○ slotID = 215 ○ lotID = 8 ○ valid = false
Actual Result	List<TicketDto>: ● TicketDto1: ○ ticketID = 14 ○ entryDate = <generated> ○ exitDate = <generated> ○ licencePlate = “12TEST123” ○ slotID = 215 ○ lotID = 8 ○ valid = false ● TicketDto2: ○ ticketID = 13 ○ entryDate = <generated> ○ exitDate = <generated> ○ licencePlate = “55ABC555” ○ slotID = 215 ○ lotID = 8 ○ valid = false
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_8 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_8

Preconditions	• Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass tickettService gemockt ist. 2. Ein TicketDtoList wird hergestellt 3. Junit Test Case ruft getAll auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	TicketDtoList mit 5 TicketDtos: • TicketDto: ○ ticketID = 0 bis 4 ○ entryDate = <generated> ○ exitDate = <generated> ○ isValid = true ○ licencePlate = “ABC1” ○ slotID = 55 ○ lotID = 5
Expected Result	TicketDtoList mit 5 TicketDtos: • TicketDto: ○ ticketID = 0 bis 4 ○ entryDate = <generated> ○ exitDate = <generated> ○ isValid = true ○ licencePlate = “ABC1” ○ slotID = 55 ○ lotID = 5
Actual Result	TicketDtoList mit 5 TicketDtos: • TicketDto: ○ ticketID = 0 bis 4 ○ entryDate = <generated> ○ exitDate = <generated> ○ isValid = true ○ licencePlate = “ABC1” ○ slotID = 55 ○ lotID = 5
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_8 (Siehe Architekturdokumentation)

VehicleController Test Cases

a) **getAll**

System Test Case ID Name	S_TC_PAU_REQ_9
Preconditions	Web Service ist gegeben: 1. Request: GET getAll() 2. Response: List<TicketDto> ○ Data “TicketDto” ■ Integer ticketID ■ Date entryDate ■ Date exitDate ■ boolean isValid ■ String licencePlate ■ Integer slotID ■ Integer lotID 3. Vehicle1 mit licencePlate = “12TEST123” ist konfiguriert: ○ licencePlate = “12TEST123” ○ vehicleType = “AUTO” ○ TicketList hat nur ein Ticket mit ticketID = 14: ■ ticketID = 14 ■ entryDate = <generated> ■ exitDate = null ■ isValid = true ■ Vehicle: Vehicle1 ■ parkSlot = 215 ■ parkingLot = TestParkingLot 4. Vehicle2 mit licencePlate = “77TEST77” ist konfiguriert: ○ licencePlate = “77TEST77” ○ vehicleType = “AUTO” ○ TicketList hat nur ein Ticket mit ticketID = 15: ■ ticketID = 15 ■ entryDate = <generated> ■ exitDate = null ■ isValid = true ■ Vehicle: Vehicle2 ■ parkSlot = 214 ■ parkingLot = TestParkingLot 5. App port number: 8080 (default) 6. System is up and running.
Test Steps	1. GET getAll() ist aufgerufen.

	2. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none
Test Data	none
Expected Result	List<VehicleDto>: ● VehicleDto1: ○ licencePlate = “123TEST123” ○ vehicleType = “AUTO” ○ ticketID = 14 ○ ● VehicleDto2: ○ licencePlate = “77TEST77” ○ vehicleType = “AUTO” ○ ticketID = 15
Actual Result	LList<VehicleDto>: ● VehicleDto1: ○ licencePlate = “123TEST123” ○ vehicleType = “AUTO” ○ ticketID = 14 ○ ● VehicleDto2: ○ licencePlate = “77TEST77” ○ vehicleType = “AUTO” ○ ticketID = 15
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_9 (Siehe Architekturdokumentation)

Component Integration Test Case ID Name	CI_TC_PAU_REQ_9
Preconditions	● Web Service Anruf ist gemockt.
Test Steps	1. Es wird sichergestellt, dass vehicleService gemockt ist. 2. Ein VehicleDtoList wird hergestellt 3. Junit Test Case ruft getAll auf. 4. Response Data werden analysiert und die erwarteten Daten werden ausgewählt.
Post-Condition	none

Test Data	VehicleDtoList mit 5 VehicleDtos: ● VehicleDto: ○ licencePlate = “ABC0” bis “ABC4” ○ vehicleType = “AUTO” ○ ticketID = 0 bis 4
Expected Result	VehicleDtoList mit 5 VehicleDtos: ● VehicleDto: ○ licencePlate = “ABC0” bis “ABC4” ○ vehicleType = “AUTO” ○ ticketID = 0 bis 4
Actual Result	VehicleDtoList mit 5 VehicleDtos: ● VehicleDto: ○ licencePlate = “ABC0” bis “ABC4” ○ vehicleType = “AUTO” ○ ticketID = 0 bis 4
Verdict (Pass/Fail)	**Pass**
Verified UC & Req. IDs	REQ_9 (Siehe Architekturdokumentation)

Chapter 3: Rückverfolgbarkeit