Test Plan and Results Document

Clinic Appointment & Patient Management System

Group B

Team Members: Halimatu Sadia Mohammed & Hanif Olayiwola

Class Being Tested:

Patient

Test Case Table (Before Coding)

Test ID	Method Being Tested	Scenario Description	Input(s)	Expected Output / Behaviour
TC1	__init__()	**Normal Case:** Create a patient with valid data	patient_id="P001" name="John Doe" age=35 contact="51234567" gender="Male"	Patient object created successfully with all attributes set correctly
TC2	__str__()	**Normal Case:** Display patient information	Patient object with: patient_id="P002" name="Sarah Smith" age=28 contact="52345678" gender="Female"	Returns formatted string: "P002 | Sarah Smith | Age: 28 | Contact: 52345678 | Gender: Female"
TC3	`__init__()`	**Edge Case:** Create patient with minimum age	patient_id="P003" name="Baby Jones" age=1 contact="53456789" gender="Female"	Patient object created successfully with age=1
TC4	__init__()	**Edge Case:** Create patient	name="Elder Smith"	Patient object

		with maximum age (120 years)	age=120
			contact="5456789 0"
			gender="Male
				created successfully with age=120

TC5	__init__()	**Invalid Case:** Create patient with a negative age	patient_id="P005"
			name="Invalid User"
			age=-5
			contact="5567890 1"
			gender="Male"
				ValueError raised or validation error message displayed

TC6	__init__()	**Invalid Case:** Create patient with non-numeric age	patient_id="P006"
			name="Test Patient"
			age="twenty"
			contact="5678901 2"
			gender="Female"
				ValueError raised when converting age to int

Test Execution Table (After Coding)

Test ID	Actual Output	Pass/Fail	Comments
TC1			
TC2			
TC3			
TC4			

TC5			

Class Being Tested:

Doctor

Test Case Table (Before Coding)

Test ID	Method Being Tested	Scenario Description	Input(s)	Expected Output / Behaviour
TC1	__init__()	Normal Case: Create doctor with valid data	doctor_id="D001", name="Dr. Jane Cooper", specialty="General Practice", available_days="Mon-Tue-Wed-Thu-Fri", start_time="08:00", end_time="17:00"	Doctor object created successfully with available_days split into list ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
TC2	is_available_on_day()	Normal Case: Check if doctor works on a working day	Doctor D001 (works Mon-Fri), date="2026-01-15" (Wednesday)	Returns True (doctor works on Wednesday)
TC3	is_within_working_hours()	Edge Case: Check time at exact start of working hours	Doctor D001 (08:00-17:00), time="08:00"	Returns True (start time is within working hours)

TC4	is_available_on_day()	Edge Case: Check if doctor works on non-working day	Doctor D002 (works Mon-Wed-Fri only), date="2026-01-16" (Thursday)	Returns False (doctor doesn't work Thursday)
TC5	is_within_working_hours()	Invalid Case: Check time outside working hours	Doctor D001 (08:00-17:00), time="18:00"	Returns False (18:00 is after end time)
TC6	__init__()	Invalid Case: Create doctor with empty available_days	doctor_id="D003", name="Dr. Test", specialty="Dentistry", available_days="", start_time="09:00", end_time="17:00"	available_days becomes empty list ["] or error

Test Execution Table (After Coding)

Test ID	Actual Output	Pass/Fail	Comments
TC1			
TC2			
TC3			
TC4			
TC5			

Class Being Tested:

Appointment

Test Case Table (Before Coding)

Test ID	Method Being Tested	Scenario Description	Input(s)	Expected Output / Behaviour
TC1	__init__()	Normal Case: Create appointment with valid data	appointment_id="A001", patient_id="P001", doctor_id="D001", date="2026-01-20", time="09:00", duration=30, department="General Consultation", purpose="Annual Checkup", status="Booked"	Appointment object created successfully with all attributes set correctly
TC2	get_end_time()	Normal Case: Calculate end time for 30-minute appointment	Appointment with: time="09:00", duration=30	Returns "09:30"
TC3	get_end_time()	Edge Case: Calculate end time that crosses hour boundary	Appointment with: time="09:45", duration=30	Returns "10:15"
TC4	get_end_time()	Edge Case: Calculate end time for long appointment (120 min)	Appointment with: time="10:00", duration=120	Returns "12:00"
TC5	__init__()	Invalid Case: Create appointment with negative duration	appointment_id="A002", patient_id="P001", doctor_id="D001", date="2026-01-20", time="10:00", duration=-30, department="Dental",	Duration=-30 stored (no validation) or error raised

			purpose="Checkup", status="Booked"	
TC6	__init__()	Invalid Case: Create appointment with non-numeric duration	appointment_id="A003", patient_id="P001", doctor_id="D001", date="2026-01-20", time="11:00", duration="thirty", department="X-Ray", purpose="Scan", status="Booked"	ValueError raised when converting duration to int

Test Execution Table (After Coding)

Test ID	Actual Output	Pass/Fail	Comments
TC1			
TC2			
TC3			
TC4			
TC5			

Class Being Tested:

ClinicManager

Test Case Table (Before Coding)

Test ID	Method Being Tested	Scenario Description	Input(s)	Expected Output / Behaviour
TC1	add_patient()	Normal Case: Add new patient to system	User inputs: name="Alice Brown", age=30, contact="57890123", gender="Female"	Patient added successfully with auto-generated ID (e.g., P006), saved to CSV, success message displayed
TC2	book_appointment()	Normal Case: Book appointment with no conflicts	patient_id="P001" (exists), doctor_id="D001" (exists), date="2026-01-20" (Wed), time="09:00", department="General", duration=30	Appointment booked successfully with ID A007, saved to CSV
TC3	slot_available()	Edge Case: Book appointment immediately after existing one (back-to-back)	Existing: D001, 2026-01-20, 09:00-09:30, New: D001, 2026-01-20, 09:30-10:00	Returns True (no overlap, appointments are back-to-back)
TC4	check_doctor_availability()	Edge Case: Check doctor availability on boundary of working hours	doctor_id="D001" (works 08:00-17:00), date="2026-01-20", time="17:00"	Returns True (17:00 equals end_time, within hours)
TC5	book_appointment()	Invalid Case: Book appointment for non-existent patient	patient_id="P999" (doesn't exist), doctor_id="D001", date="2026-01-20", time="10:00"	Error message: "Patient not found", appointment not created
TC6	slot_available()	Invalid Case: Double booking (overlapping appointment)	Existing: D001, 2026-01-20, 09:00-09:30, New: D001, 2026-01-20, 09:15-09:45	Returns False, error: "Time slot is already booked"

Test Execution Table (After Coding)

Test ID	Actual Output	Pass/Fail	Comments
TC1			
TC2			
TC3			
TC4			
TC5			