Test Protocol					
Test Case ID	**Test Case Description**	**Test Case Steps**	**Expected Result**	**Actual Result**	**Pass/Fail**
MCAL Module					
GPIO Driver					
TC_GPIO_001	Test GPIO_init	create struct that holds all pin configs and send it to DIO_init_pin(&strucct)	the pin initializes correctly	*Matches Expected Result*	Pass
TC_GPIO_002	Test GPIO_write	send port and pin and level to GPIO_write(port,pin,level)	the level is set on pin correctly	*Matches Expected Result*	Pass
TC_GPIO_003	Test GPIO_toggle	send port and pin to GPIO_togge(port,pin)	the status of pin is toggled correcty	*Matches Expected Result*	Pass
TC_GPIO_004	Test GPIO_read	send port and pin and address of variable to GPIO_read(port,pin,&value)	the status stored in variable correctly	*Matches Expected Result*	Pass
TC_GPIO_005	Test GPIO_enable_interrupt	send port and pin to GPIO_enable_interrupt(port,pin)	the interrupt enabled correctly	*Matches Expected Result*	Pass
TC_GPIO_006	Test GPIO_disable_interrupt	send port and pin to GPIO_disable_interrupt(port,pin)	the interrupt ddisabled correctly	*Matches Expected Result*	Pass
GPT Driver					
TC_GPT_001	Test GPT_init	calling the GPT_init()	All Configration Intialize Succesful	*Matches Expected Result*	Pass
TC_GPT_002	Test start_time_ms	calling the GPT_start_time_ms() and required delay	provide the required delay in milliseconds.	*Matches Expected Result*	Pass
TC_GPT_003	Test start_time_us	calling the GPT_start_time_us() and passing the required delay	provide the required delay in microseconds.	*Matches Expected Result*	Pass
TC_GPT_004	Test elabsed_time	calling the GPT_elapsed_time(), send pointer to var to store the elapsed time.	store the elapsed timer	*Matches Expected Result*	Pass
TC_GPT_005	Test remaining_time	calling the GPT_remaining_time(), send pointer to var to store the remaining ti	store the remaining time	*Matches Expected Result*	Pass
TC_GPT_006	Test enable_interrupt	calling the GPT_enable_interrupt(), and send the timer channel id	enable the interrupt.	*Matches Expected Result*	Pass
TC_GPT_007	Test disable_interrupt	calling the GPT_disable_interrupt(), and send the timer channel id	disable the interrupt.	*Matches Expected Result*	Pass
HAL Module					
Button Driver					
TC_BTN_001	Intialize Push Button	Call BUTTON_init To Intialize Push Button	All Configration Intialize Succesful	*Matches Expected Result*	Pass
TC_BTN_002	Get Push Button Status	Call BUTTON_read To Get Its Status Pressed Or Relased	Push Button Status Returned Succesful	*Matches Expected Result*	Pass
LED Driver					
TC_LED_001	Test LED_init	call LED_init	all LEDS initialized correctly	*Matches Expected Result*	Pass
TC_LED_002	Test LED_on	call LED_on and pass led id	the led turned on	*Matches Expected Result*	Pass
TC_LED_003	Test LED_off	Call LED_off and pass led id	the led turned off	*Matches Expected Result*	Pass
TC_LED_004	Test LED_toggle	Call LED_toggle and pass led id	the led toggled	*Matches Expected Result*	Pass
Application					
APP					
TC_APP_001	initialize all Hal Modules	Call led_init and button_init	all modules initializes correctly	*Matches Expected Result*	Pass
TC_APP_002	Run main Logic of application	implement main logic in super loop	app works fine and covered all known cases	*Matches Expected Result*	Pass
User Stories					
*RGB_Brightness_APP_*	Intializing all the modules	power up the system	All modules are initialized correctly	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	The RGB LED is OFF initially	first press of sw1	The Green LED is OFF	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	The Green LED will be on with a 30% duty cycle	second press of sw1	The Green LED is 30% on	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	The Green LED will be on with a 60% duty cycle	third press of sw1	The Green LED is 60% on	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	The Green LED will be on with a 90% duty cycle	fourth press of sw1	The Green LED is 90% on	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	The Green LED will be off	fifth press of sw1	The RGB LEDS are off correctly	*Matches Expected Result*	Pass
*RGB_Brightness_APP_*	On the fifth press, system state will return to st	sixth press of sw1	The RED LED is on correctly - sequence reapeted forever	*Matches Expected Result*	Pass