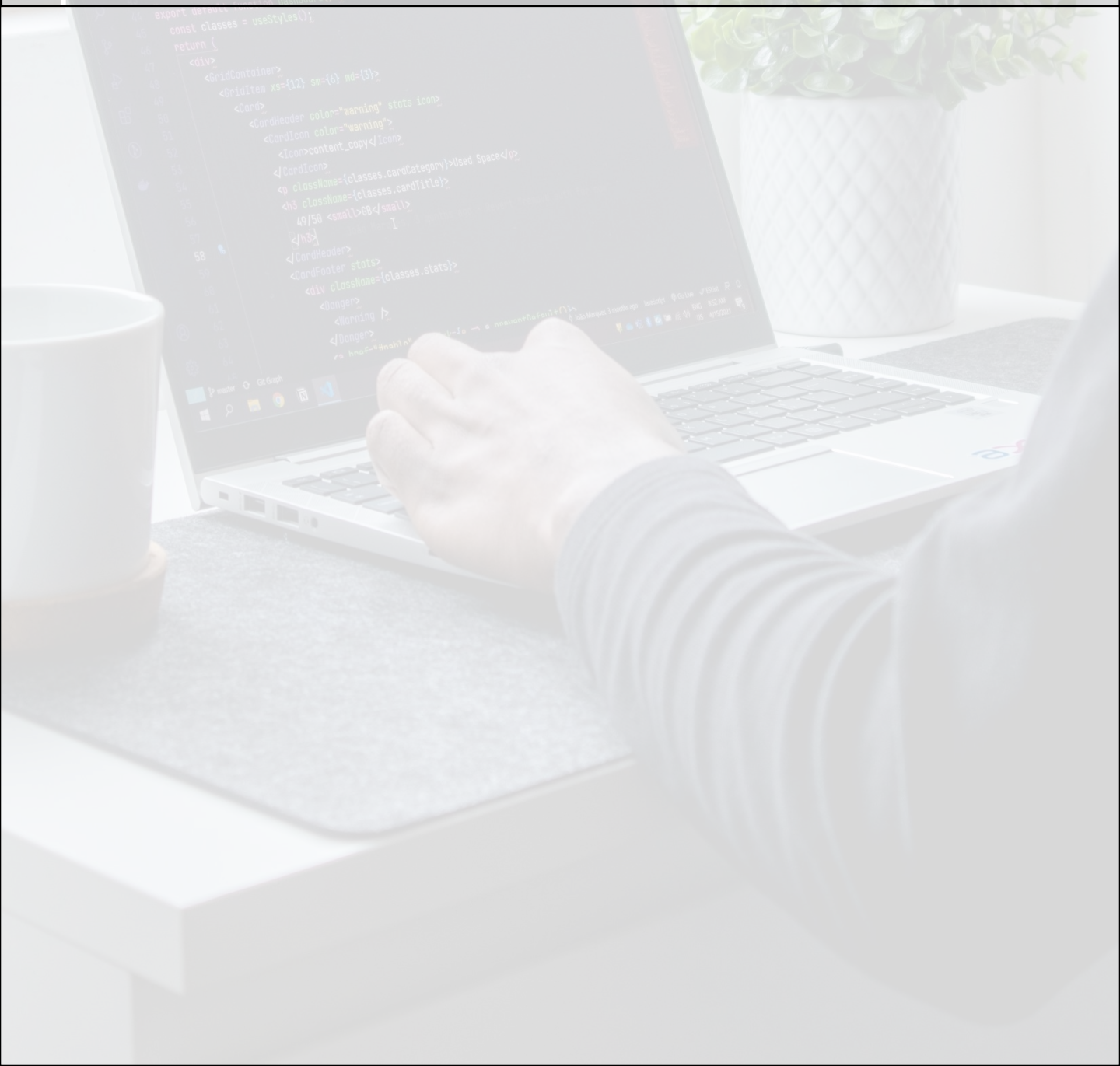
SYSTEM_REQUIREMENTS	
*	Write the LCD driver [LCD.c , LCD.h , LCD_CFG.h] The driver must be able to operate the LCD in both modes [4-bit mode , 8-bit mode] Let me choose the mode of operation in LCD_CFG.h file The LCD.c file must contain the following functions :
1	void LCD_INIT() To initialize the LCD as output component Send commands to LCD to choose mode of operation [4-bit or 8-bit]
2	void LCD_WRITE_CMD(u8 command) Use this function to send command to LCD
3	void LCD_WRITE_CHR(u8 character) Use this function to display character on LCD
4	void LCD_WRITE_STRING(u8*p) Use this function to display string on LCD
5	void LCD_WRITE_NUM(u32 number) Use this function to display numbers on LCD
6	void LCD_CLEAR() Use this function to clear the LCD screen
7	void LCD_STOP_GO_TO(u8 row , u8 col) Use this function to display in a specific row and column in the LCD
8	void LCD_STOP_WATCH_DISPLAY(u8 second , u8 minut , u8 hour) Use this function to display the time in this form (hh:mm:ss)

SYSTEM_REQUIREMENTS	
*	Write the BUZZER driver [BUZZER.c , BUZZER.h , BUZZER_CFG.h] The BUZZER.c file must contain the following functions :
1	void BUZZER_INIT(void)
	To initialize the buzzer as output component
2	void BUZZER_ON(void)
	Turning on buzzer
3	void BUZZER_OFF(void)
	Turning off buzzer
4	void BUZZER_ONCE(void)
	Generate short beep sound for one time
5	void BUZZER_TWICE(void)
	Generate short beep sound two times
6	void BUZZER_TRIPLE(void)
	Generate short beep sound three times
7	void BUZZER_LONG(void)
	Generate long beep sound for one time

SYSTEM_REQUIREMENTS

*	Write the TIMER_0 driver [TIMER_0.c , TIMER_0.h , TIMER_0_CFG.h] The TIMER.c file must contain the following functions :
1	void TIMER_0_INIT(void)
	To initialize the TIMER_0
2	void TIMER_0_SET_TIME(u32 desired_time)
	Send your desired time to this function -> 1 second
3	void TIMER_0_START(void)
	To start TIMER_0
4	void TIMER_0_STOP(void)
	To stop TIMER_0
5	ISR(TIMER_0_vect)
	Use this function to increase seconds and if seconds = 60, clear it and increase minutes by 1 and if minutes reach 60 clear it and increase hours by 1

SYSTEM_REQUIREMENTS	
*	Write the PUSH_BUTTON driver [P_B.c , P_B.h , P_B_CFG.h] The P_B.c file must contain the following functions :
1	void PUSH_BUTTON_INIT(u8 button) Make all push button pins are inputs Enable the pull up resistors for all push button pins Make PB7 pin as output Make PB7 low
2	u8 PUSH_BUTTON_READ(u8 button) Use this function to select a specific push button to read and return 0 if pressed
*	NOTE_1 : solve bouncing problem NOTE_2 : take the action if released push button



SYSTEM_REQUIREMENTS

*	Create two files in application layer : [APP.c , APP.h] These two files must contain the following functions :
1	void ASKING(void)
	Check the 4 push buttons, if any push button pressed, set a specific flag and pass the question number to CHECK_ANSWER()
2	void CHECK_ANSWER(u8 question)
	Receive the question number and switch on it from case 1 to case 30 In each case switch on the flag of the right answer from the questions If set call RIGHT_ANSWER() If clear call WRONG_ANSWER()
3	void RIGHT_ANSWER(void)
	Call BUZZER_ONCE() Display in LCD " GREAT " increase question by 1 (flag represent the current question number)
4	void WRONG_ANSWER(void)
	Call BUZZER_ONCE() Display in LCD " 1 MINUTE PENALTY " Display in LCD " LOSER " Increase minutes by 1