**MART-B - Report**

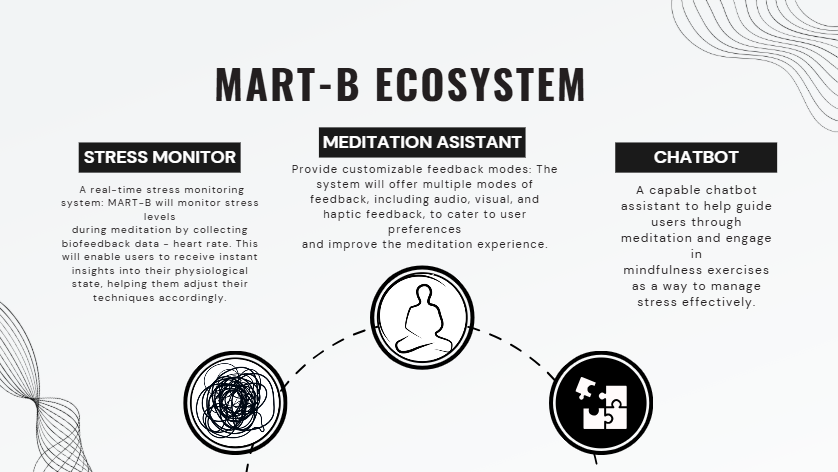
**Team:  
1. Omer Hayat – oh2122  
2. Safayet Jamil – mj3473  
3. Taha Shafiq – ts5332  
4. Bilal Awan – ba2754**

**Github Link:** <https://github.com/OmerNYU/CPE-TERM-PROJECT.git>

**Introduction**

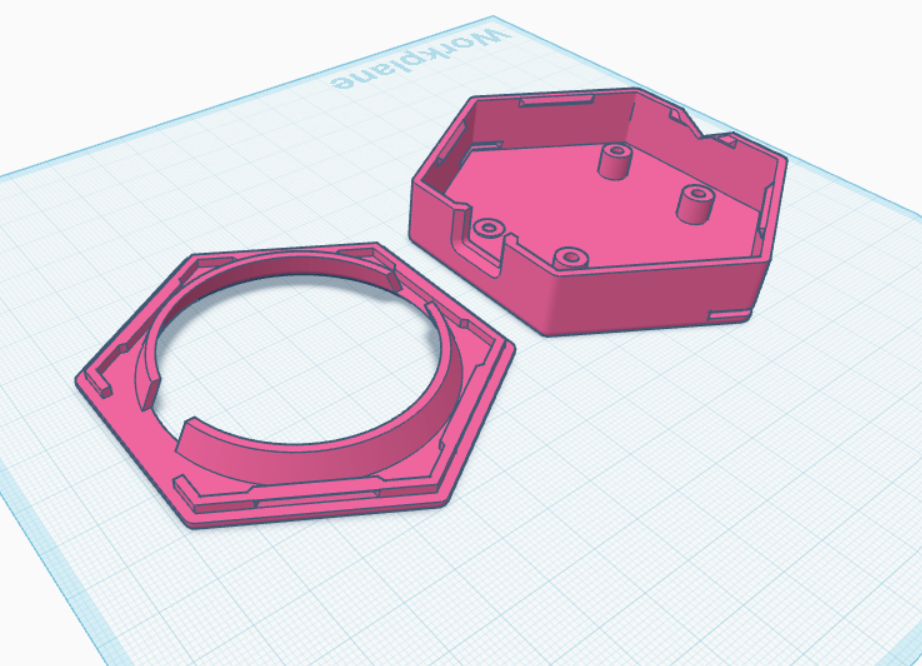
This project focuses on developing a Meditation integrated with a chatbot for holistic stress management. Our project primarily uses the Adafruit Circuit Playground for heart rate measurement and features a social chatbot that serves as both an empathetic friend and a meditation guide. This feedback loop enhances the overall meditation experience by providing personalized guidance, helping users understand their physiological state during sessions. In today’s fast-paced world, chronic stress has become a prevalent issue affecting individuals’ physical and mental well-being. According to the American Psychological Association (APA), more than 70% of adults in the United States report experiencing stress or anxiety daily (APA, 2020). Long-term exposure to stress can lead to adverse health conditions, including cardiovascular diseases, depression, and weakened immune responses (APA, 2020). As people seek ways to mitigate these effects, meditation has emerged as a widely adopted solution due to its ability to reduce stress, promote relaxation, and enhance mental clarity (Smith, 2019). We wanted to precisely tackle this issue. Real-time biofeedback offers valuable insights into a user's stress levels, enabling immediate adjustments to meditation techniques. Our project was based on ensuring user safety and offering customizable feedback options throughout the design, development, and implementation phases.

**Project Features & Development**



**Hardware Design**

1. **Adafruit Casing Design**
   * Designed a wearable casing using Tinker CAD, modifying a base design for optimal stability and usability.
   * Added slits for Velcro straps to make the casing wearable.
   * Resolved interference issues with screw joints by removing them, enhancing wire stability.
   * Adjusted wire positions by re-soldering connections to fit the casing and ensure a secure lid closure.
   * Included a triangular slit in the Tinker CAD design to facilitate wire passage.
   * Made a clipped 3D design so that we have room for future improvements.
   * The space allocated for wires was designed keeping future developments in our mind



1. **Soldering Challenges**
   * Frequent wire breaks were resolved by using heat shrink tubing for added stability, though replacing wires became necessary when the tubing complicated repairs.
   * Removed excess solder from the VBATT to 3.3V to allow proper casing assembly and also to minimize environmental interferences caused by the analog power output.
   * We soldered the device from top, but for ergonomic and user comfort purposes we shifted the solders to the side so that the device wouldn’t poke the wearer.

**Software Development**

1. **Adafruit Code**

This part of our ecosystem uses the Adafruit Circuit Playground Express and a Pulse Sensor. The software is designed to provide real-time biofeedback, offering both auditory and visual modes for stress monitoring and guided breathing for relaxation. The code is structured to ensure modularity, user interactivity, and seamless mode transitions.

### Features

The program includes the following core functionalities:

1. Mode Selection:  
   Users can select between Stress Monitoring and Meditation Assistance modes using the buttons on the Circuit Playground Express.
2. Stress Monitoring:
   * Visual Mode: LED indicators show stress levels (Green for relaxed, Yellow for moderate, Red for high stress).
   * Audio Mode: Tones are played to indicate stress levels (High-pitched tones for high stress, low-pitched tones for calmness).
3. Meditation Assistance:
   * Visual Mode: LEDs guide the user through breathing patterns by lighting up sequentially, toggling between inhale and exhale phases.
   * Audio Mode: Musical tones (based on Saregamapadhanisa) are synchronized with the breathing pattern, providing a soothing experience.
4. Heartbeat Detection:  
   Using the Pulse Sensor library, the program accurately detects beats per minute (BPM), filters erratic readings, and calculates average BPM over a specified interval.
5. Adaptive Thresholds:  
   The software dynamically calculates stress thresholds (Relaxed, Moderate) based on the user’s age, ensuring personalized feedback.
6. Error Handling:
   * Warns the user if no heartbeat is detected for more than 5 seconds.
   * Filters out unrealistic BPM readings to maintain accuracy.
7. User Input:  
   Prompts the user to input their age for personalized heart rate thresholds using the Serial Monitor.
8. Failsafe:  
   The system allows the user to reset the program and return to the main menu by pressing both buttons simultaneously.

### Software Architecture

The software is divided into several modular components for ease of readability and maintenance:

1. Setup and Initialization:
   * Initializes the Circuit Playground hardware and Pulse Sensor.
   * Configures default animations for visual feedback during startup.
2. Main Loop:
   * Handles mode selection and delegates control to specific mode loops (stressMonitorLoop, meditationAssistanceLoop).
3. Mode-Specific Loops:
   * Stress Monitoring: Tracks heartbeats, calculates BPM, and provides feedback based on stress thresholds.
   * Meditation Assistance: Guides users through breathing exercises with visual or audio cues.
4. Utility Functions:
   * lightUpAllLeds, turnOffAllLEDs, updateBreathLED, playSoothingBreathMusic, etc., encapsulate specific functionalities, ensuring clean and reusable code.

We lay our code breakdown below:

Code Breakdown for Stress Monitor and Meditation Assistant:

* setup()
  + Initializes the Arduino and prepares the device for operation
  + Starts serial communication at 115200 baud rate
  + Initializes the CircuitPlayground library
  + Configures the Pulse Sensor:
  + Sets the analog input pin
  + Establishes a threshold of 550 for beat detection
  + Initiates the Pulse Sensor
  + Executes a light animation as a startup indicator
  + Waits for simultaneous press of left and right buttons
  + Introduces a 2-second delay for button debouncing
  + Calls the chooseMode() function to select operating mode
* loop()
  + Main program loop that continuously executes
  + Checks the current program mode:
  + If mode is 1, calls stressMonitorLoop()
  + If mode is 2, calls meditationAssistanceLoop()
  + Monitors for simultaneous press of left and right buttons
  + If both buttons are pressed:
  + Introduces a 2-second delay for debouncing
  + Resets program mode, stress monitor mode, and meditation mode
  + Calls setup() to restart the program
* chooseMode()
  + Presents user with mode selection options
  + Continuously checks for button presses until a mode is selected
  + For Stress Monitor (left button):
  + Sets programMode to 1
  + Prompts user to choose between visual and audio modes
  + Waits for user input to set mode (1 for visual, 2 for audio)
  + For Meditation Assistance (right button):
  + Sets programMode to 2
  + Prompts user to choose between audio and non-audio modes
  + Waits for user input to set mode2 (1 for audio, 2 for non-audio)
  + Calls enterAge() to get user's age
  + Records the start time of the session
* enterAge()
  + Prompts user to enter their age via Serial Monitor
  + Waits for user input
  + Reads and stores the entered age
  + Calculates maximum heart rate (HRmax) using the formula: 206.9 - (0.67 \* age)
  + Determines resting heart rate (HRrest) based on age brackets
  + Prints confirmation messages to Serial Monitor
* stressMonitorLoop()
  + Manages the stress monitoring functionality
  + Tracks interval start time, monitoring start time, reading count, and BPM sum
  + Detects heartbeats using the Pulse Sensor
  + Filters out erratic BPM values (keeps 40-200 BPM range)
  + Calculates average BPM every 15 seconds
  + Determines stress levels based on calculated thresholds:
    - relaxedThreshold = HRrest + 0.15 \* (HRmax - HRrest)
    - moderateThreshold = HRrest + 0.35 \* (HRmax - HRrest)
  + Provides visual feedback using LEDs or audio feedback using tones based on stress level
  + Prints average BPM for each interval
  + Monitors total duration (60 seconds) and ends session accordingly
  + Detects absence of heartbeat for more than 5 seconds and reports it
* meditationAssistanceLoop()
  + Manages the meditation assistance functionality
  + Calculates relaxed and moderate heart rate thresholds
  + Detects heartbeats and calculates BPM
  + Provides visual feedback on stress levels using stressLights() function
  + Updates breath LED indicator using updateBreathLED() function
  + Manages breathing pattern visualization
  + Plays soothing breath music if audio mode is selected
  + Monitors for absence of heartbeat and reports if not detected for 5 seconds
* lightUpAllLeds(int red, int green, int blue)
  + Sets all LEDs on the CircuitPlayground to a specified RGB color
  + Iterates through all pixels and sets their color
  + Updates the LED strip to display the new colors
* turnOffAllLEDs()
  + Turns off all LEDs on the CircuitPlayground
  + Sets color of all pixels to black (0, 0, 0)
  + Updates the LED strip to apply the changes
* updateBreathLED()
  + Updates the LED display for breath visualization
  + Illuminates the current breath LED brightly
  + Dims the previous breath LED
  + Turns off all other LEDs in the breath range
  + Updates the LED strip to display the new pattern
* playSoothingBreathMusic()
  + Generates soothing tones synchronized with the breathing pattern
  + Determines the current tone based on breath phase (inhale/exhale) and LED position
  + Plays the appropriate tone for a duration proportional to the LED timing
  + Uses separate tone arrays for inhale and exhale phases
* lightAnimation()
  + Creates a random color animation on all LEDs
  + Generates random RGB values for each LED
  + Applies the random colors to all 10 LEDs
  + Updates the LED strip to display the new colors
  + Introduces a 1-second delay for visual effect
* stressLights(int r, int g, int b)
  + Sets the color of the first and last LEDs to indicate stress level
  + Adjusts the brightness of the LEDs based on the BRIGHTNESS constant
  + Used to provide visual feedback on current stress level in meditation mode

And below is the code itself:

1. #include <Adafruit\_CircuitPlayground.h>

2. #include <PulseSensorPlayground.h>

3.

4. // Pin and Pulse Sensor setup

5. #define PULSE\_INPUT\_PIN 7

6. //virtual pin allows us flexibility to choose the wiring

7. #define NUM\_PIXELS 10

8. #define FIRST\_BREATH\_LED 1

9. #define LAST\_BREATH\_LED 8

10. #define BRIGHTNESS 1

11. // Stress Monitor Variables

12. PulseSensorPlayground pulseSensor;

13. bool alive = false;

14. unsigned long lastAlive;

15. double HRmax, HRrest;

16. int mode = 0; // Stress Monitor:1 = Visual, 2 = Audio

17. int mode2 =0; // Meditation Assistance: 1 = Audio, 2 = Visual

18. unsigned long startTime = 0; // Stores the start time

19. const unsigned long duration = 60000; // Total duration: 60 seconds

20. const unsigned long interval = 15000; // Interval for each quarter: 15 seconds

21. int bpm;

22. float relaxedThreshold;

23. float moderateThreshold;

24.

25. // Meditation Assistance Variables

26. double moderateBPM=0; double freeBPM=0; double highBPM=0;

27. int breathLED = FIRST\_BREATH\_LED;

28. int prevBreathLED = FIRST\_BREATH\_LED;

29. bool breathToggle = false;

30. unsigned long lastBreath = 0;

31. unsigned long ledTimes[] = {500, 500, 500, 500, 500, 500, 500, 500, 500, 500};

32. int inhaleTones[] = {262, 294, 330, 349, 392, 440, 494, 523};  // Inhale tones: Saregamapadhanisa

33. int exhaleTones[] = {523, 494, 440, 392, 349, 330, 294, 262};  // Exhale tones: Sanidhapamagaresa

34.

35. // Program state

36. int programMode = -1; // -1 = Menu, 1 = Stress Monitor, 2 = Meditation Assistance

37.

38. void setup() {

39.   Serial.begin(115200);

40.   CircuitPlayground.begin();

41.

42.   // Initialize Pulse Sensor

43.   pulseSensor.analogInput(PULSE\_INPUT\_PIN);

44.   pulseSensor.setThreshold(550);

45.   pulseSensor.begin();

46.

47.   // Default light animation on startup

48.   lightAnimation();

49.

50.   while (!(CircuitPlayground.leftButton()&& CircuitPlayground.rightButton())){

51.     continue;

52.   }

53.   delay(2000);

54.   chooseMode();

55. }

56.

57. void loop() {

58.   if (programMode == 1) {

59.     stressMonitorLoop();

60.   } else if (programMode == 2) {

61.     meditationAssistanceLoop();

62.   }

63.   if(CircuitPlayground.leftButton()&& CircuitPlayground.rightButton()){

64.     delay(2000);

65.     programMode=-1;

66.     mode=0;

67.     mode2=0;

68.     setup();

69.   }

70. }

71.

72. void chooseMode() {

73.   Serial.println("Welcome to Mart-B!");

74.   Serial.println("Press Left Button for Stress Monitor, Right Button for Meditation Assistance.");

75.   while (programMode == -1) {

76.     if (CircuitPlayground.leftButton()) {

77.       programMode = 1;

78.       Serial.println("Stress Monitor Selected.");

79.       Serial.println("Which functionality of Stress Monitor are you going to use today?");

80.       Serial.println("Left Button for Visual, Right button for Auditory");

81.       delay(2000);

82.       while(mode==0){

83.         if(CircuitPlayground.leftButton()){

84.           mode = 1;

85.           Serial.println("You selected the visual mode, yay!");

86.         } else if (CircuitPlayground.rightButton()){

87.           mode = 2;

88.           Serial.println("You selected the audio mode, yay!");

89.         }

90.       }

91.       enterAge();

92.       startTime = millis();

93.     } else if (CircuitPlayground.rightButton()) {

94.       programMode = 2;

95.       Serial.println("Meditation Assistance Selected. There are 2 modes for Meditation Assistance. Do you want to do it in peace(right button for no music) or beats?(left button for with music)");

96.       delay(2000);

97.       while(mode2==0){

98.         if(CircuitPlayground.leftButton()){

99.           mode2 = 1;

100.           Serial.println("You selected the audio mode, yay!");

101.         } else if (CircuitPlayground.rightButton()){

102.           mode2 = 2;

103.           Serial.println("You selected the non-audio mode, Good choice!");

104.         }

105.       }

106.       enterAge();

107.       startTime = millis();

108.     }

109.   }

110. }

111.

112. void enterAge() {

113.   Serial.println("Please enter your age via the Serial Monitor:");

114.   while (Serial.available() == 0) {

115.     // Wait for input

116.   }

117.   int age = Serial.parseInt();

118.   Serial.print("Age entered: ");

119.   Serial.println(age);

120.   Serial.println("It begins!");

121.   // Calculate HRmax and HRrest based on age

122.   HRmax = 206.9 - (0.67 \* age); //https://www.heartonline.org.au/resources/calculators/target-heart-rate-calculator

123.   if (age <= 1) {

124.     HRrest = 120;

125.   } else if (age <= 3) {

126.     HRrest = 110;

127.   } else if (age <= 5) {

128.     HRrest = 100;

129.   } else if (age <= 12) {

130.     HRrest = 90;

131.   } else if (age <= 19) {

132.     HRrest = 75;

133.   } else if (age <= 64) {

134.     HRrest = 72;

135.   } else {

136.     HRrest = 70;

137.   }

138. }

139.

140. void stressMonitorLoop() {

141.   static unsigned long intervalStart = millis(); // Tracks the start of the current 15-second interval

142.   static unsigned long monitoringStart = millis(); // Tracks the start of the 60-second duration

143.   static int readingCount = 0; // Count of BPM readings in the current interval

144.   static int bpmSum = 0; // Sum of BPM readings in the current interval

145.   unsigned long currentTime = millis();

146.

147.   if (pulseSensor.sawStartOfBeat()) {

148.     alive = true;

149.     lastAlive = currentTime;

150.

151.     bpm = pulseSensor.getBeatsPerMinute();

152.

153.     if (bpm >= 40 && bpm <= 200) { // Filter out erratic values

154.       Serial.println(bpm);

155.       bpmSum += bpm;

156.       readingCount++;

157.     }

158.   }

159.

160.   // Check if 15 seconds have passed

161.   if (currentTime - intervalStart >= interval) {

162.     intervalStart = currentTime;

163.

164.     if (readingCount > 0) {

165.       int avgBpm = bpmSum / readingCount;

166.       bpmSum = 0; // Reset for the next interval

167.       readingCount = 0;

168.

169.       relaxedThreshold = HRrest + 0.15 \* (HRmax - HRrest);

170.       moderateThreshold = HRrest + 0.35 \* (HRmax - HRrest);

171.

172.       if (mode == 1) { // Visual mode

173.         if (avgBpm < relaxedThreshold) {

174.           lightUpAllLeds(0, 255 \* BRIGHTNESS, 0); // Green

175.         } else if (avgBpm >= relaxedThreshold && avgBpm <= moderateThreshold) {

176.           lightUpAllLeds(255 \* BRIGHTNESS, 255 \* BRIGHTNESS, 0); // Yellow

177.         } else {

178.           lightUpAllLeds(255 \* BRIGHTNESS, 0, 0); // Red

179.         }

180.       } else if (mode == 2) { // Audio mode

181.         if (avgBpm > moderateThreshold) {

182.           CircuitPlayground.playTone(880, 500); // High pitch for stress

183.         } else {

184.           CircuitPlayground.playTone(440, 500); // Low pitch for calm

185.         }

186.       }

187.

188.       Serial.print("Average BPM for this interval: ");

189.       Serial.println(avgBpm);

190.     }

191.   }

192.   // Check if the 60-second duration has passed

193.   if (currentTime - monitoringStart >= duration) {

194.     Serial.println("Stress monitoring complete.");

195.     delay(7000);

196.     turnOffAllLEDs();

197.     programMode = -1; // Return to the menu

198.     bpmSum=0;

199.     readingCount = 0;

200.   }

201.

202.   // Handle no heartbeat detected for more than 5 seconds

203.   if (currentTime - lastAlive > 5000 && alive) {

204.     alive = false;

205.     Serial.println("No heartbeat detected last 5 seconds!");

206.   }

207. }

208.

209. void meditationAssistanceLoop() {

210.   unsigned long timeNow = millis();

211.   float relaxedThreshold = HRrest + 0.15 \* (HRmax - HRrest);

212.   float moderateThreshold = HRrest + 0.35 \* (HRmax - HRrest);

213.   // Update the pulse sensor reading and check if a beat was detected

214.   if (pulseSensor.sawStartOfBeat()) {

215.     alive = true;

216.     lastAlive = timeNow;

217.

218.     // Calculate BPM and set stress coherence rating

219.     int bpm = pulseSensor.getBeatsPerMinute();

220.     if (bpm >= 200) { // Handle false sound values

221.       Serial.println("Erratic value discarded!");

222.     } else {

223.       Serial.print("\*\*\* Heartbeat Detected \*\*\* BPM: ");

224.       Serial.println(bpm);

225.     }

226.

227.     if (bpm < relaxedThreshold) {

228.       stressLights(0,255,0);

229.     } else if (bpm >= relaxedThreshold && bpm <= moderateThreshold) {

230.       stressLights(0,0,255);  // Moderate BPM indicates moderate stress

231.     } else {

232.       stressLights(255,0,0);  // High BPM indicates high stress

233.     }

234.

235.     // Move the breath LED

236.     updateBreathLED();

237.   }

238.

239.   // Handle no heartbeat detected for more than 5 seconds

240.   if (timeNow - lastAlive > 5000 && alive) {

241.     alive = false;

242.     Serial.println("No heartbeat detected in the last five seconds");

243.   }

244.

245.   // Update LED positions based on breath timing

246.   unsigned long timeSince = timeNow - lastBreath;

247.

248.   if (timeSince >= ledTimes[breathLED]) {

249.     prevBreathLED = breathLED;

250.     breathLED += breathToggle ? -1 : 1;

251.

252.     if (breathLED < FIRST\_BREATH\_LED || breathLED > LAST\_BREATH\_LED) {

253.       breathToggle = !breathToggle; // Toggle between inhale and exhale

254.       breathLED = constrain(breathLED, FIRST\_BREATH\_LED, LAST\_BREATH\_LED);

255.     }

256.     lastBreath = timeNow;

257.   }

258.

259.   // Play tones for the current breath phase

260.   if (mode2==1 && alive){

261.   playSoothingBreathMusic();

262.   }

263. }

264.

265. void lightUpAllLeds(int red, int green, int blue) {

266.   for (int i = 0; i < NUM\_PIXELS; i++) {

267.     CircuitPlayground.setPixelColor(i, red, green, blue);

268.   }

269.   CircuitPlayground.strip.show();

270. }

271.

272. void turnOffAllLEDs() {

273.   for (int i = 0; i < NUM\_PIXELS; i++) {

274.     CircuitPlayground.setPixelColor(i, 0, 0, 0);

275.   }

276.   CircuitPlayground.strip.show();

277. }

278.

279. void updateBreathLED() {

280.   for (int i = FIRST\_BREATH\_LED; i <= LAST\_BREATH\_LED; i++) {

281.     if (i == breathLED) {

282.       CircuitPlayground.setPixelColor(i, 200, 200, 200);

283.     } else if (i == prevBreathLED) {

284.       CircuitPlayground.setPixelColor(i, 50, 50, 50);

285.     } else {

286.       CircuitPlayground.setPixelColor(i, 0, 0, 0);

287.     }

288.   }

289.   CircuitPlayground.strip.show();

290. }

291.

292. void playSoothingBreathMusic() {

293.   unsigned long timeNow = millis();

294.   unsigned long timeSince = timeNow - lastBreath;

295.

296.   // Determine tone based on breath phase and current LED

297.   int toneIndex = breathLED - FIRST\_BREATH\_LED; // Index of the current tone in the array

298.   int toneFreq = breathToggle ? exhaleTones[toneIndex] : inhaleTones[toneIndex];

299.

300.   // Play the tone if within the current LED's timing

301.   if (timeSince <= ledTimes[breathLED]) {

302.     CircuitPlayground.playTone(toneFreq, ledTimes[breathLED] / 10); // Synchronized tones

303.   }

304. }

305.

306. void lightAnimation() {

307.   for (int i = 0; i < 10; i++) {

308.     CircuitPlayground.setPixelColor(i, random(0, 255\*BRIGHTNESS), random(0, 255\*BRIGHTNESS), random(0, 255\*BRIGHTNESS));

309.   }

310.   CircuitPlayground.strip.show();

311.   delay(1000);

312. }

313.

314. void stressLights(int r, int g, int b) {

315.     // Gradually reduce fade rate for pulse indicator LEDs

316.   // Apply fade effect to pulse LEDs

317.   CircuitPlayground.strip.setPixelColor(0,   r / 255\*BRIGHTNESS,   g / 255\*BRIGHTNESS,   b / 255\*BRIGHTNESS);

318.   CircuitPlayground.strip.setPixelColor(9,   r / 255\*BRIGHTNESS,   g / 255\*BRIGHTNESS,   b / 255\*BRIGHTNESS);

319. }

320.

### Innovations

1. Real-Time Stress Feedback:  
   By integrating heart rate thresholds with LED and audio outputs, the system provides immediate and intuitive feedback.
2. Personalization:  
   Age-based HRmax and HRrest calculations allow the system to adapt to the user’s physiological characteristics.
3. Multi-Modal Assistance:  
   The project supports auditory and visual feedback, catering to different user preferences and enhancing accessibility.
4. Dynamic Breath Control:  
   The meditation assistance mode uses synchronized LED animations and tonal variations to guide users through inhale and exhale phases.
5. Interactive Menu System:  
   The use of buttons for mode selection and Serial Monitor for age input ensures a user-friendly interface.

**Chatbot Development**

1. **Features and API Selection**
   * Designed a dual-function chatbot:
     + **Empathetic Friend**: Assisted with stress relief.
     + **Meditation Instructor**: Guided meditation routines.
   * Selected the Gemini v1.5 Flash Model API for its open-source compatibility.
2. **Programming Language**
   * Used Python for its familiarity, versatility, and ease of implementation.
3. **Training and Deployment**
   * Leveraged Google AI Studio (Vertex AI) for chatbot engineering and training.
   * Hosted the chatbot using Streamlit, a library for building interactive web apps.
4. **Challenges**
   * Initial difficulties in identifying APIs and platforms for chatbot hosting were resolved by adopting Streamlit.

**Chatbot Code:**

1. import streamlit as st # type: ignore

2. import vertexai

3. from vertexai.generative\_models import GenerativeModel

4.

5. # Initialize Vertex AI

6. PROJECT\_ID = "unified-atom-441618-q6"

7. vertexai.init(project=PROJECT\_ID, location="us-central1")

8.

9. # Set up the Gemini model with a calming persona

10. model = GenerativeModel(

11.     "gemini-1.5-flash-002",

12.     system\_instruction="You are an empathetic friend and a calming meditation instructor."

13. )

14.

15. # Define chatbot functions

16.

17. def generate\_response(prompt):

18.     try:

19.         response = model.generate\_content(prompt)

20.         return response.text

21.     except Exception as e:

22.         st.error(f"Error generating response: {e}")

23.         return "I'm having trouble understanding. Can you try rephrasing?"

24.

25. def generate\_meditation\_script(type="deep breathing", duration=5, user\_name="friend"):

26.     prompt = (f"Create a {duration}-minute guided meditation focusing on {type}. "

27.               f"The script should be calming and reassuring, emphasizing relaxation and presence. "

28.               f"Address the user as '{user\_name}'.")

29.     return generate\_response(prompt)

30.

31. def provide\_stress\_advice():

32.     prompt = "Provide advice on managing stress and anxiety in a gentle, understanding tone."

33.     return generate\_response(prompt)

34.

35. def generate\_empathetic\_response(emotion):

36.     prompt = f"Offer a supportive and empathetic response for someone feeling {emotion}."

37.     return generate\_response(prompt)

38.

39. # Streamlit UI

40. st.title("Meditation & Wellness Chatbot")

41. st.write("Welcome! I'm here to help with meditation, stress advice, or just to chat.")

42.

43. # Initialize chat history and session state variables

44. if "chat\_history" not in st.session\_state:

45.     st.session\_state.chat\_history = []

46.

47. # Handle user input and response generation

48. user\_input = st.text\_input("How can I help you today?", key="user\_input")

49.

50. if st.button("Send") and user\_input:

51.     # Check user intent and respond accordingly

52.     if "meditation" in user\_input.lower():

53.         # Prompt for meditation details if not already set

54.         if "meditation\_type" not in st.session\_state:

55.             st.session\_state.meditation\_type = st.text\_input("What type of meditation? (e.g., deep breathing, body scan)")

56.             st.session\_state.meditation\_duration = st.number\_input("Duration in minutes", min\_value=1, max\_value=60, value=5)

57.         else:

58.             # Generate meditation script

59.             response = generate\_meditation\_script(type=st.session\_state.meditation\_type, duration=int(st.session\_state.meditation\_duration))

60.             st.session\_state.chat\_history.append(("User", user\_input))

61.             st.session\_state.chat\_history.append(("Chatbot", response))

62.             # Clear temporary variables

63.             del st.session\_state.meditation\_type

64.             del st.session\_state.meditation\_duration

65.

66.     elif "stress" in user\_input.lower():

67.         # Generate stress advice response

68.         response = provide\_stress\_advice()

69.         st.session\_state.chat\_history.append(("User", user\_input))

70.         st.session\_state.chat\_history.append(("Chatbot", response))

71.

72.     elif "feeling" in user\_input.lower():

73.         # Extract emotion from user input

74.         emotion = user\_input.lower().split("feeling")[-1].strip()

75.         if emotion:

76.             response = generate\_empathetic\_response(emotion)

77.             st.session\_state.chat\_history.append(("User", user\_input))

78.             st.session\_state.chat\_history.append(("Chatbot", response))

79.

80.     else:

81.         # General response for other input

82.         response = generate\_response(user\_input)

83.         st.session\_state.chat\_history.append(("User", user\_input))

84.         st.session\_state.chat\_history.append(("Chatbot", response))

85.

86. # Display conversation history

87. for sender, message in st.session\_state.chat\_history:

88.     if sender == "User":

89.         st.write(f"\*\*You:\*\* {message}")

90.     else:

91.         st.write(f"\*\*Chatbot:\*\* {message}")

92.

93.

**1. Overview**

The code implements a chatbot using **Streamlit** for the UI and **Google's Vertex AI** to provide generative responses. The chatbot focuses on wellness by offering meditation guidance, stress management advice, and empathetic conversations. It uses the **Gemini model** for content generation with a calming and empathetic persona.

**2. Explanation of Code Components**

**2.1 Imports and Initialization**

1. import streamlit as st # type: ignore

2. import vertexai

3. from vertexai.generative\_models import GenerativeModel

4.

* **Streamlit** provides the interactive web-based UI.
* **Vertex AI and GenerativeModel** handle generative AI functionalities. The code sets up the **Gemini model** with a specific persona.

1. PROJECT\_ID = "unified-atom-441618-q6"

2. vertexai.init(project=PROJECT\_ID, location="us-central1")

3.

* **PROJECT\_ID and location** specify the Google Cloud project and region. This ensures communication with the appropriate Vertex AI instance.

**2.2 Setting Up the Model**

1. model = GenerativeModel(

2. "gemini-1.5-flash-002",

3. system\_instruction="You are an empathetic friend and a calming meditation instructor."

4. )

5.

* **GenerativeModel** initializes the Gemini model for generating content. The system\_instruction defines its persona as empathetic and calming, tailoring its responses.

**2.3 Chatbot Functions**

The key functions are:

* **generate\_response(prompt)**: Generates general responses for user prompts using the Gemini model.
* **generate\_meditation\_script**: Creates guided meditation scripts by specifying type (e.g., "deep breathing") and duration.
* **provide\_stress\_advice**: Offers stress management tips in a gentle tone.
* **generate\_empathetic\_response**: Responds empathetically to specific emotions extracted from user input.

Each function uses the generate\_response helper function to retrieve AI-generated text.

**2.4 Streamlit UI and User Interaction**

* **Title and Introduction**: The UI starts with a title and brief introduction.
* **Chat History Management**: Uses **Streamlit's session state** to maintain conversation history persistently across user interactions.

if "chat\_history" not in st.session\_state:

st.session\_state.chat\_history = []

* **User Input Handling**: Captures user input via a text field (st.text\_input) and a button to trigger processing.
* **Intent Recognition**: Analyzes user input to determine the intent:
  + **Meditation Request**: Prompts users to specify meditation type and duration before generating a meditation script.
  + **Stress Advice Request**: Calls the provide\_stress\_advice function.
  + **Emotion Support**: Detects emotional expressions to generate empathetic responses.
  + **Fallback Handling**: Generates a general response for unrecognized inputs.
* **Chat Display**: Displays user-bot conversation history in an intuitive format using Streamlit components.

**3. Strengths**

1. **Modular Design**:
   * Functions are separated by functionality, ensuring code is modular and easy to extend.
2. **Personalization**:
   * Meditation scripts and empathetic responses are tailored to user preferences (e.g., type of meditation, emotion).
3. **Streamlit Integration**:
   * Provides a clean and user-friendly interface for the chatbot.
4. **Error Handling**:
   * Includes basic error handling in generate\_response to address potential model failures.

**4. Areas for Improvement**

1. **Input Validation**:
   * The code does not validate user inputs (e.g., meditation types, emotions). This can lead to incorrect or suboptimal responses.
   * **Solution**: Add input validation (e.g., predefined options for meditation types).
2. **Error Feedback**:
   * While errors are caught, the feedback to users is generic.
   * **Solution**: Display meaningful error messages or retry suggestions.
3. **Session State Management**:
   * Temporary variables (meditation\_type, meditation\_duration) are cleared manually, which is error-prone.
   * **Solution**: Use a structured approach to manage session state variables.
4. **Performance**:
   * Generating responses for every interaction may increase latency.
   * **Solution**: Cache frequently requested responses (e.g., standard stress advice).
5. **Security**:
   * The PROJECT\_ID is hardcoded, which could expose the project to unauthorized access.
   * **Solution**: Store sensitive credentials in environment variables or a secure configuration file.
6. **UI Enhancements**:
   * The UI does not provide clear feedback when awaiting responses.
   * **Solution**: Add a loading spinner or message for better user experience.

**5. Possible Extensions**

1. **Broader Functionality**:
   * Include features like mood tracking or gratitude journaling.
2. **Advanced Personalization**:
   * Allow users to set persistent preferences for tone, meditation type, or duration.
3. **Analytics**:
   * Track interaction trends to refine chatbot responses over time.
4. **Voice Input/Output**:
   * Support voice commands for a more immersive experience.

**Website Code:**

**Index.html:**

1. <!DOCTYPE HTML>

2.

3. <html>

4.     <head>

5.         <!-- The title of the webpage as it will appear on the browser tab -->

6.         <title>MART-B</title>

7.

8.         <!-- Specifies the character encoding for the document -->

9.         <meta charset="utf-8" />

10.

11.         <!-- Ensures the webpage is responsive and scales properly on all devices -->

12.         <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />

13.

14.         <!-- Links the main CSS stylesheet for styling the page -->

15.         <link rel="stylesheet" href="assets/css/main.css" />

16.

17.         <!-- Links an alternative CSS file for when JavaScript is disabled -->

18.         <noscript><link rel="stylesheet" href="assets/css/noscript.css" /></noscript>

19.     </head>

20.     <body class="is-preload">

21.         <!-- 'is-preload' is likely a class used for applying initial styles before the page is fully loaded -->

22.

23.         <!-- Sidebar section -->

24.         <section id="sidebar">

25.             <div class="inner">

26.                 <nav>

27.                     <ul>

28.                         <!-- Links to the intro and 'What We Do' sections of the page -->

29.                         <li><a href="#intro">Welcome</a></li>

30.                         <li><a href="#one">What We Do</a></li>

31.                     </ul>

32.                 </nav>

33.             </div>

34.         </section>

35.

36.         <!-- Wrapper that contains the main content -->

37.         <div id="wrapper">

38.

39.             <!-- Introduction section -->

40.             <section id="intro" class="wrapper style1 fullscreen fade-up">

41.                 <div class="inner">

42.                     <!-- The main title of the webpage -->

43.                     <h1>MART-B</h1>

44.

45.                     <!-- Brief description of the project -->

46.                     <p>Welcome to the home of Meditation Assistant with Real Time Biofeedback also known as MART-B, a stress monitoring system with a meditation assistant to help you relax and focus.</p>

47.

48.                     <!-- Action button to navigate to the next section -->

49.                     <ul class="actions">

50.                         <li><a href="#one" class="button scrolly">Learn more</a></li>

51.                     </ul>

52.                 </div>

53.             </section>

54.

55.             <!-- Section explaining the purpose and features of the project -->

56.             <section id="one" class="wrapper style2 spotlights">

57.                 <section>

58.                     <!-- Image for visual appeal -->

59.                     <a href="#" class="image"><img src="images/pic01.jpg" alt="" data-position="center center" /></a>

60.

61.                     <div class="content">

62.                         <div class="inner">

63.                             <!-- Subheading: Why the project exists -->

64.                             <h2>Why?</h2>

65.

66.                             <!-- Explanation of the problem the project addresses -->

67.                             <p>In the fast-paced lifestyle of the present, chronic stress has become extremely common, affecting the physical and mental wellbeing of many individuals. To deal with this, many people turn to meditation as a solution. However, without instant real-time feedback, it can be difficult for people to gauge their stress levels and the effectiveness of their meditation routines. </p>

68.

69.                             <!-- Action button linking to more information -->

70.                             <ul class="actions">

71.                                 <li><a href="info.html" class="button">Learn more</a></li>

72.                             </ul>

73.                         </div>

74.                     </div>

75.                 </section>

76.

77.                 <section>

78.                     <!-- Another image for visual interest -->

79.                     <a href="#" class="image"><img src="images/pic02.jpg" alt="" data-position="top center" /></a>

80.

81.                     <div class="content">

82.                         <div class="inner">

83.                             <!-- Subheading: The solution offered by the project -->

84.                             <h2>Our Solution</h2>

85.

86.                             <!-- Description of how the project solves the problem -->

87.                             <p>Our project aims to address this problem by developing a stress monitor using the Adafruit Circuit Playground Classic. This project will provide users with real-time biofeedback on their stress levels during meditation sessions utilizing heart rate, enabling them to adjust their techniques and achieve a more effective and personalized meditation experience. </p>

88.

89.                             <!-- Action button linking to more details -->

90.                             <ul class="actions">

91.                                 <li><a href="generic.html" class="button">Learn more</a></li>

92.                             </ul>

93.                         </div>

94.                     </div>

95.                 </section>

96.

97.                 <section>

98.                     <!-- Another visual element -->

99.                     <a href="#" class="image"><img src="images/pic03.jpg" alt="" data-position="25% 25%" /></a>

100.

101.                     <div class="content">

102.                         <div class="inner">

103.                             <!-- Subheading: Meditation guidance -->

104.                             <h2>Your Meditation Guide</h2>

105.

106.                             <!-- Details about the chatbot feature -->

107.                             <p>Using Gemini's API, we have made a chatbot to assist users in their meditation routines. Not just that, the chatbot is there as a friend to listen and help you refocus and relax.</p>

108.

109.                             <!-- Disclaimer for users -->

110.                             <p>Disclaimer: It is not a substitute for licensed therapist.</p>

111.

112.                             <!-- Action button linking to the chatbot application -->

113.                             <ul class="actions">

114.                                 <li><a href="https://gembot-yrht3mksggg7ucnsncjmkp.streamlit.app/" class="button">Learn more</a></li>

115.                             </ul>

116.                         </div>

117.                     </div>

118.                 </section>

119.             </section>

120.

121.         <!-- Footer section -->

122.         <footer id="footer" class="wrapper style1-alt">

123.             <div class="inner">

124.                 <ul class="menu">

125.                     <!-- Footer content: Project name -->

126.                     <li>MART-B</li>

127.                 </ul>

128.             </div>

129.         </footer>

130.

131.         <!-- JavaScript files for interactive features -->

132.         <script src="assets/js/jquery.min.js"></script>

133.         <script src="assets/js/jquery.scrollex.min.js"></script>

134.         <script src="assets/js/jquery.scrolly.min.js"></script>

135.         <script src="assets/js/browser.min.js"></script>

136.         <script src="assets/js/breakpoints.min.js"></script>

137.         <script src="assets/js/util.js"></script>

138.         <script src="assets/js/main.js"></script>

139.     </body>

140. </html>

141.

142.

**Info.html:**

1. <!DOCTYPE HTML>

2. <!-- HTML5 document structure begins -->

3.

4. <html>

5.     <head>

6.         <title>Deep Dive</title> <!-- Page title displayed on the browser tab -->

7.         <meta charset="utf-8" /> <!-- Specifies the character encoding as UTF-8 -->

8.         <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" /> <!-- Responsive viewport for mobile devices -->

9.         <link rel="stylesheet" href="assets/css/main.css" /> <!-- Links the main stylesheet for styling -->

10.         <noscript><link rel="stylesheet" href="assets/css/noscript.css" /></noscript> <!-- Fallback CSS for users with JavaScript disabled -->

11.     </head>

12.     <body class="is-preload"> <!-- Body with preload class applied for initial animations -->

13.

14.         <!-- Header -->

15.             <header id="header"> <!-- Website header -->

16.                 <a href="index.html" class="title">Back to Home</a> <!-- Link to navigate back to the homepage -->

17.                 <nav> <!-- Navigation menu -->

18.                     <ul>

19.                         <li><a href="index.html">Home</a></li> <!-- Navigation item: Home -->

20.                         <li><a href="info.html" class="active">Info</a></li> <!-- Navigation item: Info (currently active) -->

21.                         <li><a href="generic.html">Monitor Stress</a></li> <!-- Navigation item: Monitor Stress -->

22.                         <li><a href="https://gembot-yrht3mksggg7ucnsncjmkp.streamlit.app/">Chatbot</a></li> <!-- Navigation item: External Chatbot link -->

23.                     </ul>

24.                 </nav>

25.             </header>

26.

27.         <!-- Wrapper -->

28.             <div id="wrapper"> <!-- Central container for content -->

29.

30.                 <!-- Main -->

31.                     <section id="main" class="wrapper"> <!-- Main content area -->

32.                         <div class="inner"> <!-- Inner container for padding -->

33.                             <h1 class="major">Deep Dive</h1> <!-- Main heading -->

34.                             <p>Stress is an unavoidable part of modern life...</p> <!-- Brief overview of stress and its impacts -->

35.                             <p>The following statistics highlight how widespread high-stress levels have become:</p>

36.                             <ol>

37.                                 <li>K-12 Students: In the United States, 45% of high school students report feeling stressed "all the time" or "most of the time" due to schoolwork.</li>

38.                                 <li>College Students: The American College Health Association (ACHA) found that 77% of college students experienced some form of psychological distress, either moderate or severe.</li>

39.                                 <li>Mental Health Impact: During the 2020-2021 school year, more than 60% of college students met the criteria for at least one mental health problem.</li>

40.                                 <li>Stressful Life Events: Three out of four students reported experiencing at least one stressful life event in the past year, with more than 20% experiencing six or more stressful events.</li>

41.                                 <li>Sleep and Stress: Students who sleep six or fewer hours a night have lower GPAs compared to those who get eight or more hours of sleep.</li>

42.                                 <li>Financial Stress: 24% of students in the United States are stressed about their future and finding a job after graduation.</li>

43.                                 <li>Loneliness: 54% of college students reported experiencing loneliness.</li>

44.                                 <li>Suicidal Behavior: 30% of college students exhibited suicidal behavior.</li>

45.                                 <li>Anxiety and Depression: 35% of college students reported being diagnosed with anxiety, and 27% reported experiencing depression.</li>

46.                             </ol>

47.

48.                         </div>

49.                     </section>

50.

51.             </div>

52.

53.         <!-- Footer -->

54.             <footer id="footer" class="wrapper alt"> <!-- Website footer -->

55.                 <div class="inner">

56.                     <ul class="menu">

57.                         <li>MART-B</li> <!-- Branding or site reference -->

58.                     </ul>

59.                 </div>

60.             </footer>

61.

62.         <!-- Scripts -->

63.             <script src="assets/js/jquery.min.js"></script> <!-- Core jQuery library -->

64.             <script src="assets/js/jquery.scrollex.min.js"></script> <!-- Plugin for scrolling animations -->

65.             <script src="assets/js/jquery.scrolly.min.js"></script> <!-- Plugin for smooth scrolling -->

66.             <script src="assets/js/browser.min.js"></script> <!-- Browser compatibility handling -->

67.             <script src="assets/js/breakpoints.min.js"></script> <!-- Script for responsive breakpoints -->

68.             <script src="assets/js/util.js"></script> <!-- Utility functions -->

69.             <script src="assets/js/main.js"></script> <!-- Main JavaScript file -->

70.     </body>

71. </html>

72.

73.

**Generic.html:**

1. <!DOCTYPE HTML>

2. <!-- HTML5 document structure begins -->

3.

4. <html>

5.     <head>

6.         <title>STRESS MONITOR</title> <!-- Page title displayed on the browser tab -->

7.         <meta charset="utf-8" /> <!-- Specifies the character encoding as UTF-8 -->

8.         <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" /> <!-- Responsive viewport for mobile devices -->

9.         <link rel="stylesheet" href="assets/css/main.css" /> <!-- Links the main stylesheet for styling -->

10.         <noscript><link rel="stylesheet" href="assets/css/noscript.css" /></noscript> <!-- Fallback CSS for users with JavaScript disabled -->

11.     </head>

12.     <body class="is-preload"> <!-- Body with preload class applied for initial animations -->

13.

14.         <!-- Header -->

15.             <header id="header"> <!-- Website header -->

16.                 <a href="index.html" class="title">Back to Home</a> <!-- Link to navigate back to the homepage -->

17.                 <nav> <!-- Navigation menu -->

18.                     <ul>

19.                         <li><a href="index.html">Home</a></li> <!-- Navigation item: Home -->

20.                         <li><a href="generic.html" class="active">Monitor Stress</a></li> <!-- Navigation item: Monitor Stress (currently active) -->

21.                         <li><a href="https://gembot-yrht3mksggg7ucnsncjmkp.streamlit.app/">Chatbot</a></li> <!-- Navigation item: External Chatbot link -->

22.                         <li><a href="info.html">Info</a></li> <!-- Navigation item: Info -->

23.                     </ul>

24.                 </nav>

25.             </header>

26.

27.         <!-- Wrapper -->

28.             <div id="wrapper"> <!-- Central container for content -->

29.

30.                 <!-- Main -->

31.                     <section id="main" class="wrapper"> <!-- Main content area -->

32.                         <div class="inner"> <!-- Inner container for padding -->

33.                             <h1 class="major">User Guide</h1> <!-- Main heading -->

34.                             <h2>Please read the guide below to operate the device:</h2> <!-- Subheading -->

35.                             <p>Welcome to MART-B! Your go-to ecosystem  after a long day of work to refocus and relax.</p> <!-- Introduction to the product -->

36.

37.                             <!-- Instructions on the device's functionalities -->

38.                             <ol>

39.                                 <li>Stress Monitor:</li> <!-- Details of stress monitoring modes -->

40.                                 <ul>

41.                                     <li>a. Visual Mode: The lights on the Adafruit will update every 15 seconds to indicate your stress levels. Red - Stressed; Yellow - Tense; Green - Relaxed</li>

42.                                     <li>b. Audio Mode: The device beeps and the frequency of beeping will change to reflect your stress levels. High Frequency - Stressed; Low Frequency - Relaxed</li>

43.                                 </ul>

44.                                 <li>Meditation Assistance:</li> <!-- Details of meditation assistance modes -->

45.                                 <ul>

46.                                     <li>a. Visual Mode: Sync your breathing (inhale-exhale) with the lights on the device</li>

47.                                     <li>b. Audio Mode: Sync your breathing to the beeping. Switch between inhale-exhale when the frequency changes.</li>

48.                                 </ul>

49.                             </ol>

50.

51.                             <p>To get started, press both buttons on the Adafruit together! Then, follow the instructions on the serial monitor to relax. Happy Relaxing! </p>

52.                         </div>

53.                     </section>

54.

55.             </div>

56.

57.         <!-- Footer -->

58.             <footer id="footer" class="wrapper alt"> <!-- Website footer -->

59.                 <div class="inner">

60.                     <ul class="menu">

61.                         <li>MART-B</li> <!-- Branding or site reference -->

62.                     </ul>

63.                 </div>

64.             </footer>

65.

66.         <!-- Scripts -->

67.             <script src="assets/js/jquery.min.js"></script> <!-- Core jQuery library -->

68.             <script src="assets/js/jquery.scrollex.min.js"></script> <!-- Plugin for scrolling animations -->

69.             <script src="assets/js/jquery.scrolly.min.js"></script> <!-- Plugin for smooth scrolling -->

70.             <script src="assets/js/browser.min.js"></script> <!-- Browser compatibility handling -->

71.             <script src="assets/js/breakpoints.min.js"></script> <!-- Script for responsive breakpoints -->

72.             <script src="assets/js/util.js"></script> <!-- Utility functions -->

73.             <script src="assets/js/main.js"></script> <!-- Main JavaScript file -->

74.     </body>

75. </html>

76.

77.

**Explanation:**

**1. Overview of index.html**

The index.html file is the **homepage** of the MART-B website. It introduces users to the platform, explaining its purpose and features. The structure includes a sidebar navigation menu, a main introduction section, and additional content on the project's goals and solutions.

**Code Explanation**

**Head Section**

1. <head>

2. <title>MART-B</title>

3. <meta charset="utf-8" />

4. <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />

5. <link rel="stylesheet" href="assets/css/main.css" />

6. <noscript><link rel="stylesheet" href="assets/css/noscript.css" /></noscript>

7. </head>

8. Defines the page title (MART-B) that appears on the browser tab.

9. Sets character encoding to UTF-8 for compatibility with various languages.

10. Ensures mobile responsiveness using the viewport meta tag.

11. Links external stylesheets for styling:

12. main.css for the primary design.

13. noscript.css for fallback styling when JavaScript is disabled.

14.

**Sidebar**

<section id="sidebar">

<div class="inner">

<nav>

<ul>

<li><a href="#intro">Welcome</a></li>

<li><a href="#one">What We Do</a></li>

</ul>

</nav>

</div>

</section>

* Provides a **navigation menu** for quick access to sections within the page.
* Uses <a href="#intro"> and <a href="#one"> for **internal links** to the "Welcome" and "What We Do" sections.

**Introduction Section**

1. <section id="intro" class="wrapper style1 fullscreen fade-up">

2. <div class="inner">

3. <h1>MART-B</h1>

4. <p>Welcome to the home of Meditation Assistant with Real Time Biofeedback...</p>

5. <ul class="actions">

6. <li><a href="#one" class="button scrolly">Learn more</a></li>

7. </ul>

8. </div>

9. </section>

10.

* Introduces the project with a title (<h1>) and description (<p>).
* A button with the class scrolly provides smooth scrolling to the "What We Do" section.

**Spotlight Sections**

1. <section id="one" class="wrapper style2 spotlights">

2. <!-- Subsections go here -->

3. </section>

4.

* Contains multiple subsections explaining:
  1. The **problem** MART-B addresses.
  2. MART-B's **solution** using the Adafruit Circuit Playground.
  3. Integration with the Gemini chatbot for **meditation guidance**.

Each subsection includes an image (<a class="image">), a title (<h2>), descriptive text (<p>), and a button linking to additional content.

**Footer**

1. <footer id="footer" class="wrapper style1-alt">

2. <div class="inner">

3. <ul class="menu">

4. <li>MART-B</li>

5. </ul>

6. </div>

7. </footer>

8.

* Displays the project name at the bottom of the page.

**Scripts**

<script src="assets/js/jquery.min.js"></script>

<script src="assets/js/main.js"></script>

* Includes external JavaScript files for interactivity and animations.

**2. Overview of info.html**

The info.html page provides a **detailed look at stress statistics**, which highlight the relevance of MART-B.

**Code Explanation**

**Header and Navigation**

Similar to index.html, it includes:

* A header with a navigation menu linking to the homepage, the chatbot, and the user guide.

**Main Content**

1. <section id="main" class="wrapper">

2. <div class="inner">

3. <h1 class="major">Deep Dive</h1>

4. <p>Stress is an unavoidable part of modern life...</p>

5. <ol>

6. <li>K-12 Students: 45% of high school students...</li>

7. <li>College Students: 77% of college students...</li>

8. <li>Suicidal Behavior: 30% of college students...</li>

9. </ol>

10. </div>

11. </section>

12.

* The <h1> tag introduces the topic: "Deep Dive."
* Provides textual insights into the prevalence of stress.
* Uses an <ol> element for structured, easy-to-read statistics.

**Footer and Scripts**

Identical to index.html.

**3. Overview of generic.html**

The generic.html file serves as the **user guide** for MART-B, explaining how to operate the device and use its features.

**Code Explanation**

**Main Content**

1. <section id="main" class="wrapper">

2. <div class="inner">

3. <h1 class="major">User Guide</h1>

4. <h2>Please read the guide below to operate the device:</h2>

5. <p>Welcome to MART-B!...</p>

6. <ol>

7. <li>Stress Monitor:</li>

8. <ul>

9. <li>Visual Mode: Lights on the Adafruit...</li>

10. <li>Audio Mode: Beeping frequency...</li>

11. </ul>

12. <li>Meditation Assistance:</li>

13. <ul>

14. <li>Visual Mode: Sync breathing with lights...</li>

15. <li>Audio Mode: Sync breathing with beeping...</li>

16. </ul>

17. </ol>

18. <p>To get started, press both buttons...</p>

19. </div>

20. </section>

21.

* **Instructions**: The user guide is divided into two primary modes:
  1. **Stress Monitor**: Uses lights or beeps to indicate stress levels.
  2. **Meditation Assistance**: Guides users to sync their breathing with visual/auditory cues.
* Each mode is detailed using a nested ordered (<ol>) and unordered list (<ul>).

**Footer and Scripts**

Similar to the other files.

**4. Key Commonalities**

* **Structure**: All files follow a consistent layout with headers, navigation menus, main content, and footers.
* **Styling and Interactivity**: Reliance on external CSS and JavaScript for responsive design and animations.
* **Purpose**:
  + index.html: Introduction and overview.
  + info.html: Background and statistics.
  + generic.html: User instructions.

**Results and Evaluation**

**Hardware Performance**

* **Casing Stability**: The redesigned casing provided enhanced stability and wearability, ensuring the device remained comfortably and securely in place during user sessions.
* **Durability Improvements**: Re-soldered connections addressed frequent wire breaks, resulting in significantly more robust hardware performance.

**Software Functionality**

* **Stress Monitoring Accuracy**:
  + We evaluated the device's heart rate detection by comparing its readings to commercial heart rate monitors.
  + The system accurately detected heart rates within the 40-200 BPM range, with erratic values outside this range successfully filtered out to improve reliability.
  + Simplified logic-based calculations further enhanced the consistency of heart rate readings, addressing the limitations of the sensor.
* **Real-Time Feedback**:
  + Timely feedback was achieved through optimized code, including refined interval calculations and minimized unnecessary computations in the main loop. This reduced latency between heart rate changes and feedback delivery via LEDs or audio tones.
* **Meditation Assistance Effectiveness**:
  + User testing revealed that the visual LED guidance for breathing exercises was intuitive and effective.
  + The audio feedback, designed using tones based on the **Saregamapadhanisa scale**, received positive feedback for its calming influence.
* **Challenges Addressed**:
  + **Heart Rate Sensor Sensitivity**: Initial inconsistencies in pulse sensor readings were mitigated by adjusting sensitivity thresholds and implementing a robust beat detection algorithm.
  + **Age-Based Threshold Calculations**: Fixed thresholds were replaced with age-based calculations for maximum heart rate (HRmax) and resting heart rate (HRrest), improving the accuracy of stress assessments.

**Chatbot Evaluation**

* **Response Accuracy and Relevance**:
  + The chatbot demonstrated high accuracy in recognizing user intents related to stress relief and meditation guidance.
  + Some responses for complex queries were identified as areas needing contextual improvement.
* **Meditation Script Generation**:
  + User feedback indicated that the meditation scripts generated by the chatbot were calming and well-structured.
  + For longer sessions (>15 minutes), occasional repetition was observed, suggesting the need for enhanced script variety.
* **Debugging and Optimization**:
  + **API Integration**: Inconsistent response times with the Gemini v1.5 Flash Model API were resolved by implementing robust error handling and retry mechanisms.
  + **Memory Management**: Increasing memory usage due to growing conversation history was optimized by introducing periodic cleanup and efficient conversation history handling.
  + **Cross-Platform Compatibility**: Issues with CSS rendering on mobile devices and JavaScript functionality on certain browsers were addressed, ensuring consistent performance across platforms.

**Chatbot Hosting and User Access**

* **Streamlit Hosting**:
  + The chatbot was successfully hosted on Streamlit, offering seamless user access to its functionalities.
  + This ensured an intuitive and user-friendly interface for stress management and meditation guidance.

**Conclusion**

The rigorous evaluation and debugging process for both hardware and software significantly enhanced the reliability, functionality, and user experience of MART-B. By addressing key challenges and optimizing features, the system now provides users with accurate real-time biofeedback, customizable meditation assistance, and an engaging chatbot interface to effectively manage stress.

**Conclusion and Future Work**

The project successfully delivered a wearable stress measurement device integrated with a chatbot, addressing stress management through real-time monitoring and interactive guidance. Future improvements include:

* **Battery-Powered and Portable Design**
  + One of the primary enhancements for future iterations of the project is to integrate an inbuilt rechargeable battery. This addition would eliminate the dependency on external power sources, making the system more portable and convenient for users. A compact lithium-polymer battery could be used along with a power management circuit to ensure safe charging and efficient power usage. This would allow the device to function seamlessly in various environments, enabling users to carry it during outdoor activities or travel without needing constant access to a power outlet.
* **External Microphone for Voice Interaction**
  + To enhance user accessibility, an external microphone interface could be integrated into the system, allowing users to interact with the chatbot via voice commands instead of typing. This feature would make the system more user-friendly, especially for individuals who may find typing on small devices inconvenient. Advanced speech recognition libraries like Google's Speech-to-Text or offline alternatives could be explored to ensure accurate voice processing. Such a feature could also pave the way for incorporating conversational AI, making the system capable of engaging in more natural and intuitive interactions.
* **Improved Heart Rate Sensor**
  + Upgrading to a high-precision heart rate sensor is another crucial improvement for future versions of the project. By using a sensor capable of accurately measuring interbeat intervals (IBI), the system could deliver more reliable and detailed insights into stress levels. With better IBI accuracy, the device could implement advanced algorithms such as heart rate variability (HRV) analysis, which is a proven metric for stress detection. This enhancement would significantly increase the system’s effectiveness in monitoring and managing stress, making it more suitable for clinical or professional wellness applications.
* **Implementing a Multithreaded Processor for Synchronized Experiences**
  + In future iterations, we plan to upgrade the device’s processor to a multithreaded model, significantly improving its performance and functionality. This enhancement will enable seamless synchronization between music and light patterns, eliminating the need for complex and time-consuming calculations to align the two. By running separate threads for audio and visual cues, the device can deliver perfectly timed, immersive experiences that enhance the meditation process. This improvement will also allow for the addition of more sophisticated features, such as adaptive light and sound patterns that adjust in real-time to the user’s stress levels or breathing patterns.
* **Incorporating a Heat Sensor for Environmental Awareness**
  + We aim to integrate a heat sensor capable of detecting temperature changes in the user’s immediate environment. This feature will allow the device to monitor external conditions that might influence stress or comfort levels. For instance, detecting a rise in temperature could trigger recommendations for cooling techniques or adjustments to meditation settings. This added functionality would make the device more responsive and adaptive, further personalizing the user experience.
* **Listening to Breathing Rates Using External Microphones or Respiratory Belts**
  + Another area for future development is the ability to monitor breathing rates using an external microphone or a respiratory belt. By capturing audio signals of the user’s breath or the physical expansion and contraction of their chest, the device could provide real-time feedback on breathing patterns. This data could be used to fine-tune the meditation guidance, ensuring it aligns with the user’s natural breathing rhythm. Such a feature would make the device more intuitive and effective, particularly for users looking to deepen their meditation practice.
* **Enhancing the Chatbot API for Greater Functionality**
  + To improve the chatbot's versatility and responsiveness, we plan to upgrade its API. This enhancement would enable smoother integration with the meditation device and external platforms, allowing the chatbot to access more data points and provide more nuanced advice. For example, integrating stress data from the device could allow the chatbot to tailor its suggestions based on real-time feedback. Improved API functionality would also open doors for third-party developers to expand the chatbot’s capabilities, fostering innovation and adaptability.
* **Redesigning the Chatbot UI for User-Friendliness**
  + An improved user interface (UI) is another priority for future work. The goal is to create a more intuitive, visually appealing design that enhances user interaction. By incorporating clear navigation, engaging visuals, and accessible features, we aim to make the chatbot experience as seamless as possible. Additional features, such as dark mode, customizable themes, and voice interaction, could further elevate the usability of the chatbot, making it more engaging and accessible to a wider audience.
* **Transition to a Paid Platform for Enhanced Functionality**
  + As part of future work, we aim to transition from using Streamlit to a paid platform that offers more robust and scalable features. A paid platform would provide advanced capabilities, such as better hosting, faster load times, enhanced security, and support for complex functionalities. This move would ensure that our interface is more reliable, professional, and capable of handling a larger user base with minimal downtime.
* **Integrating Interface Image Generation for Meditation Demonstration**
  + To make the system more user-friendly, we plan to incorporate interface-based image generation that visually demonstrates meditation techniques. These generated visuals could guide users through breathing exercises, postures, or mindfulness techniques in an intuitive and engaging way. The addition of these images would cater to visual learners and make the system more inclusive, ensuring users can easily follow along without needing prior meditation experience.
* **Code Optimization and Modularity**
  + We aim to focus on optimizing the codebase to enhance system performance and maintainability. By adopting modular programming principles, the system would be divided into self-contained, reusable components, making it easier to debug, update, and extend. Optimized code would reduce resource consumption, ensuring the device runs efficiently even as new features are added. This improvement would also streamline collaboration, allowing developers to work on individual modules without disrupting the overall system.
* **Enhanced Error Handling Mechanisms**
  + Improved error handling is a priority for future development. The system will incorporate detailed logging and dynamic error detection methods to identify issues proactively. Clear and user-friendly error messages will guide users in resolving problems, reducing frustration. Additionally, automated recovery features could reset specific components or suggest troubleshooting steps to minimize disruptions in functionality. This robust error handling would make the system more reliable and user-centric.
* **Customization for Heart Rate Settings**
  + To give users more control, we plan to add options for setting custom heart rate ranges. This feature would allow users to personalize their meditation experience based on their fitness levels or specific needs. For instance, users could configure the device to focus on maintaining a resting heart rate or adapting breathing patterns to achieve a target zone. Such customization would enhance the device’s versatility, making it suitable for a broader audience.
* **Startup Lighting for Improved Interaction**
  + A startup lighting sequence will be introduced to indicate that the device is initializing. This feature will enhance the user experience by providing immediate visual feedback that the system is operational. The lighting can also include calming colors to set the tone for meditation, ensuring users are engaged from the moment they turn on the device.
* **Error-Indicating Lighting for Clarity**
  + Lighting patterns will also be employed to signify errors. For instance, blinking red lights could alert users to issues such as sensor malfunctions or failed connectivity. This visual feedback will make troubleshooting more intuitive, especially in cases where detailed error messages might not be accessible.
* **Custom Breathing Cycles for Personalized Meditation**
  + In future updates, we plan to introduce custom breathing cycle functionality, allowing users to create and save personalized breathing patterns. This feature will let users tailor their meditation sessions to suit specific goals, such as relaxation, focus, or energy. For example, users could adjust the inhale, hold, and exhale durations to align with their preferences or guided techniques they follow. Custom breathing cycles will make the device more versatile and adaptable to individual needs, enhancing its effectiveness across different user profiles.
* **Customizable Resting Heart Rates**
  + We aim to add functionality that allows users to set custom resting heart rate ranges. This will cater to individuals with varying fitness levels or unique cardiovascular needs, such as athletes or users with medical conditions. By enabling personalized resting heart rate settings, the device can provide more accurate feedback and tailored recommendations, ensuring that it aligns with each user's physical profile.
* **Enhanced User Profiles for Better Personalization**
  + The system will be upgraded to collect additional user information, such as BMI, height, and body weight. This data will be used to refine the device’s algorithms for heart rate and stress monitoring, ensuring the values are more personalized and accurate. For example, heart rate targets for relaxation or fitness will be adjusted based on the user’s body composition, making the device more effective in achieving its goals. Collecting this data also opens the door to advanced features like fitness tracking and health monitoring in future versions.
* **Simultaneous Audio and Visual Guidance with Thread Optimization**
  + A key area of improvement is enabling simultaneous audio and visual guidance during meditation sessions. Currently limited by a single-thread processor, we plan to optimize thread usage to allow synchronized output without compromising performance. Users will experience perfectly timed audio cues alongside light patterns, enhancing the immersion and effectiveness of the meditation experience. This improvement will make the device more seamless and intuitive, especially for users who rely on multiple sensory inputs during relaxation.
* **Creating a Holistic and Adaptive System**
  + By implementing custom breathing cycles, personalizing heart rate settings, and incorporating additional user data, the device will become a more comprehensive tool for stress management and meditation. Enhanced synchronization of audio and visual outputs will further elevate the user experience. These advancements reflect our commitment to delivering a highly personalized and adaptable system that meets the diverse needs of its users.

**Reflection and Learning**

Our journey in developing MART-B, a Meditation Assistant with Real-Time Biofeedback, has been filled with valuable insights and learning experiences. This project has significantly enhanced our understanding of hardware-software integration, problem-solving strategies, and the iterative nature of product development.

**Hardware Integration Challenges**

**Sensor Limitations and Adaptations**

We encountered several challenges related to the heart rate sensor's accuracy and reliability. This experience taught us the importance of:

- Thorough research and testing of components before integration

- Developing robust error-handling mechanisms to manage unreliable sensor data

- Implementing adaptive algorithms to compensate for hardware limitations

**Ergonomic Design Considerations**

The process of designing a wearable device highlighted the critical balance between functionality and user comfort:

- We learned to prioritize user experience in hardware design, moving from a top-mounted soldering approach to a side-mounted one for better ergonomics

- The iterative design process of our 3D-printed casing emphasized the importance of prototyping and user feedback

**Software Development Insights**

**Modular Code Architecture**

Developing the software for MART-B reinforced the value of modular programming:

- We created separate functions for different functionalities, enhancing code readability and maintainability

- This approach allowed for easier debugging and feature additions throughout the development process

**Real-Time Data Processing**

Working with real-time biofeedback data presented unique challenges:

- We gained experience in filtering and processing live sensor data to provide meaningful feedback

- The project improved our skills in developing responsive systems that can adapt to user inputs and physiological changes in real-time

**Interdisciplinary Collaboration**

The project underscored the importance of cross-functional teamwork:

- We learned to effectively communicate technical concepts between hardware and software team members

- The integration of the chatbot with the hardware device taught us about API usage and cloud service integration

**User-Centric Design Approach**

Throughout the development process, we maintained a strong focus on user needs:

- We incorporated multiple feedback modes (visual and audio) to cater to different user preferences

- The project enhanced our understanding of creating intuitive user interfaces for both hardware and software components

**Ethical Considerations in Health Tech**

Developing a stress management tool brought ethical considerations to the forefront:

- We learned about the importance of clear disclaimers and setting appropriate user expectations

- The project increased our awareness of data privacy concerns in health-related applications

**Continuous Learning and Adaptation**

The dynamic nature of the project emphasized the importance of continuous learning:

- We had to quickly adapt to new technologies, such as the Gemini API for chatbot development

- The project improved our ability to research and implement solutions to unforeseen challenges

**Future Development Perspectives**

This project has opened our eyes to various possibilities for future enhancements:

- We identified potential areas for improvement, such as incorporating additional physiological metrics for more comprehensive stress assessment

- The experience has inspired ideas for expanding the project's scope, possibly integrating machine learning for personalized stress management strategies

In conclusion, the development of MART-B has been a comprehensive learning experience, enhancing our technical skills, project management abilities, and understanding of user-centered design in health technology. These insights will undoubtedly prove valuable in our future endeavors in the field of interactive and assistive technologies.

**References:**

1. Codecademy

2. FreeCodeCamp

3. Programming with Mosh

4. SuperSimpleDev

5. Corey Schafer

6. Tech with Tim

7. SentDex

8. RealPython

9. Australian Heart Foundation

10. American Psychology Association

11. Adafruit Documentation