Athens University of Economics and Business
Digital Innovation and Entrepreneurship 2026

Business Plan of Startup

Real-time stress measurement and automatic adjustment of lighting & music — with no manual intervention.

Team Members

Spilios Dimakopoulos Student ID: 8220035

Stavros Vlachos Student ID: 8220019

Instructor: Stavros Lounis

Athens, 2026

Contents

Chapter 1

Presentation of the Business Idea

1.1 Problem Description

Stress and burnout are among the greatest public-health risks of the modern era [1]. Students, employees, and young professionals face high levels of pressure every day — without having the right tools to recognize it in time.

The problem is not just that stress exists; it is that most people realize it only *after* it has already affected their performance, sleep, and mood. At the same time, smart-home devices are experiencing huge growth but remain exclusively manual.

The two central market gaps:

- Lack of a reliable, real-time stress-measurement tool for the average user
- Inability to automatically adapt the environment based on physiological state

1.2 Customer Types & Market Research

1.2.1 Target Customer

People aged **18–35** who:

- Experience daily stress and anxiety (students, young employees, professionals)
- Are tech-savvy and comfortable with technology (early adopters)
- Are looking for a passive, non-intrusive wellness solution in their space
- Care about mental health but don't want extra cognitive load

From **Year 2** onward, the second customer type is the **corporate customer (B2B)**: companies investing in their employees' mental health (employee wellness). The B2B customer purchases 10+ unit packages at a discount plus a corporate subscription of €12/user/month. A detailed presentation of the two segments appears in Ch. 5 (§ Revenue Streams by Customer Type).

1.2.2 Customer Jobs, Pains & Gains

Customer Jobs	Customer Pains	Customer Gains
Reduce stress with no effort	**[Critical]** Slow reaction	**[Required]** Reliable measurement + automatic operation
Improve sleep	**[Critical]** Lack of tools	**[Expected]** Setup < 10 minutes
Investing in health (social)	**[Important]** Manual intervention	**[Desired]** AI insights & personalization
Bodily self-awareness	**[Important]** Fragmented solutions	**[Unexpected]** Privacy-first / EU data
Managing mental health	**[Moderate]** Privacy concerns	**[Desired]** Transparent data use

Table 1.1: Customer Jobs, Pains & Gains Analysis

1.2.3 Primary Research Results

An online survey was conducted with **12 respondents** via Google Forms (students & employees, May 2026). A full breakdown per question appears in Appendix B; below are the key findings in chart form.

1.3 Market Size

Market	Interpretation	Size
TAM	Global stress-management + wearables + IoT wellness market	**~$20+ bn**
SAM	European consumer wellness IoT market, ages 18–35	**~€1.5–2 bn**
SOM	GR + DE + NL, Year 1–3, ~400–5,000 users (0.01–0.05% of tech-savvy 18–35-year-olds across the three markets)	**€69K–€650K**

Table 1.2: TAM / SAM / SOM Analysis

Sources: [2], [3], [4], [5], [6]

Note: SOM is estimated bottom-up based on projected sales (conservative–base scenario) and is consistent with the financial projections in Ch. 6 (€69K Y1 — €650K Y3).

1.4 Product Description — How It Works

DigiNova MoodLight is a complete **B2C IoT stress-management system** that measures the user's biometric data in real time and automatically adapts the environment — lighting and music — with no manual intervention.

1.4.1 System Flow

1. **Measurement:** The smart ring continuously measures HR/HRV via the MAX30102
2. **Processing:** Bluetooth → MoodBox → Cloud: Stress Score 0–100 based on a personal baseline
3. **Intervention:** Score > threshold → warm lighting + ambient music within 15–30 seconds
4. **Feedback:** Automatic reset + logging in the app

1.4.2 Hardware Bundle

- **Smart Ring:** HR/HRV + skin temperature, BLE, 5–7-day battery life
- **Smart Bulbs:** RGB WiFi bulbs — automatic color/intensity change based on Stress Score
- **LED Strip:** Ambient lighting behind furniture / around a screen
- **MoodBox:** Central hub/smart speaker — processing + sending data to the cloud

1.5 Technology & Stack

Sensors & Hardware	Cloud & Software
MAX30102: PPG for HR, SpO2 — RMSSD for HRV stress estimation [7]	Cloud backend: Stress Score + storage on EU servers (GDPR)
Skin temperature: secondary signal	Mobile App (Android/iOS): analytics + real-time Stress Score
Arduino Uno + ESP32 (WiFi): prototype hub	LLM API (Claude / GPT-4o): AI assistant for the Premium tier
DFPlayer Mini + SD card: on-device ambient sound	Spotify Web API: automatic music control (Premium)

Table 1.3: MoodLight Technology Stack

1.6 Competition & Competitive Advantage

Feature	Oura Ring [ ]	Philips Hue [ ]	Spotify	Calm App	MoodLight
HR/HRV measurement	✓	✗	✗	✗	✓
Smart lighting	✗	✓	✗	✗	✓
Music playback	✗	✗	✓	✓	✓
Automatic intervention	✗	✗	✗	✗	✓
Unified system	✗	✗	✗	✗	✓
Privacy-first / EU data	∼	∼	✗	✗	✓
Price	€299–499	€150–300	€10.99/mo	€69.99/yr	**€89.99 + €15/mo**

Table 1.4: Comparative Competitive Analysis

1.7 Revenue Streams by Customer Type

MoodLight generates revenue through two channels: a one-off sale of the HW bundle (€89.99, B2C) and a recurring Premium subscription (€15/month), with B2B expansion starting in Year 2. A detailed presentation appears in Ch. 5.

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	**Gemini**: establishing the idea, Arduino factors, hardware choice (ring / wristband / watch) **Claude**: visualization of the survey CSV (Python script), architecture diagram (Fig. 1.7)
Stavros Vlachos	8220019	**Claude**: competitive analysis (feature comparison table)

Table 1.5: AI Assistants used — Ch. 1

Chapter 2

Design Prototype (Interactive Mockups)

2.1 Mockup Tool

The interactive design prototype was developed in HTML/CSS with a dark-mode UI, following Material Design 3 guidelines for Android.

> **Figma Wireframes:**
> https://www.figma.com/proto/YBdfdGWaMMWnEhVJzW0Lur/Untitled

Figure 2.1: Splash / Index Screen

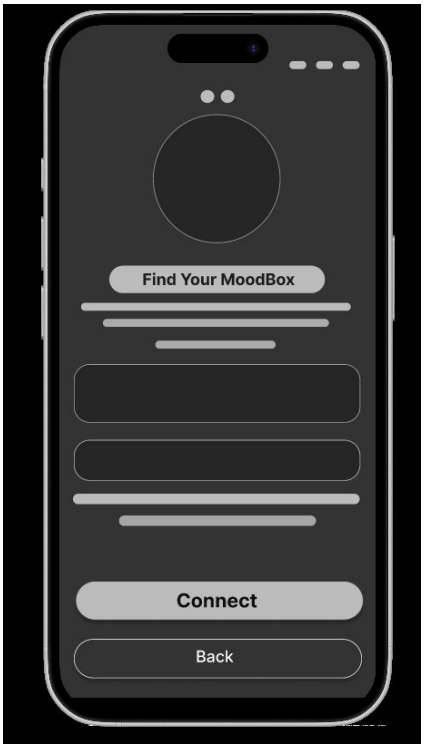

Figure 2.2: Onboarding — Connecting MoodBox

Figure 2.3: MoodBox Connected — Calibration

2.2 UI/UX Design Principles

MoodLight's design system follows four central principles. **Minimalism:** every screen exposes only one primary indicator (Stress Score or a chart), reducing cognitive load for users who are already experiencing stress. **Dark-mode first:** the dark background reduces eye strain during night-time use and reinforces the color coding (green/yellow/red) through high contrast (WCAG AA). **Affordance:** the circular gauge taps into the mental model of a "vehicle dashboard" — the user

instantly perceives whether they are "in the red" without reading a number. **Progressive disclosure:** Premium features (HRV Trends, AI Assistant) are visible but locked, creating an incentive to upgrade without burdening the Free user.

2.3 Screen Descriptions

2.3.1 Screen 1 — Onboarding & Setup

Step-by-step instructions for connecting the MAX30102/Smart Ring via Bluetooth (<10 minutes), calibration, and GDPR consent: "We only store your Stress Score, not your raw biometric data." The linear step-by-step guide (instead of free exploration) applies the guided-onboarding principle: the user doesn't need to decide what to do first — each screen has a single CTA. The GDPR statement appears during onboarding (not in settings) precisely because the research (Ch. 8) identified data protection as the users' main "pain".

2.3.2 Screen 2 — Dashboard (Home)

Real-time Stress Score (0–100) shown as an animated gauge. Color coding: green (0–40), yellow (41–64), red (65–100). Manual override available. Choosing a gauge instead of a plain number allows at-a-glance assessment (glanceability), crucial for users who are already under elevated stress and don't want cognitive burden. The animated color transition acts as a passive notification: the user doesn't need to actively open the app.

2.3.3 Screen 3 — HRV Trends & Analytics

A 7/30-day time-series chart with stress peaks and correlation with sleep. **Premium tier only**. Hiding the detailed analysis at the Free tier is not merely a business choice — it also reflects the progressive-disclosure principle: the new user is not overwhelmed with data before understanding the core functionality. The 7/30-day chart was chosen over a daily view because stress follows weekly cycles (workdays vs. weekend).

2.3.4 Screen 4 — AI Assistant (Premium)

A natural-language chat interface with personalized answers based on HRV. A weekly AI summary every Sunday. The chat UI was chosen because natural language reduces the fear of "using it wrong" and appeals to non-technical users. The weekly summary (Sunday) leverages the natural mental review of the week, increasing the perceived value of the Premium subscription.

2.3.5 Screen 5 — Settings

Control of threshold, lighting, music (Spotify / ambient), notifications, and GDPR export/delete. The threshold setting is exposed in Settings (not on the dashboard) because it's a choice the user makes once and rarely changes — placing it on the main screen would create accidental taps. The GDPR export/delete feature directly implements the "right to erasure" under GDPR Art. 17.

2.3.6 Screen 6 — Premium (Upgrade)

An upgrade screen with a Free/Premium comparison, plan selection (Monthly / Annual), and a call-to-action: "Start free 7-day trial." The side-by-side comparison (Free vs Premium) applies

the anchoring principle: the user first sees what they're missing on Free and then evaluates the price. "Start free 7-day trial" instead of an upfront payment reduces the psychological resistance to purchase, targeting the core pain: "I want to try it before I commit."

Figure 2.4: Dashboard **Figure 2.5:** HRV Trends **Figure 2.6:** MoodAI Chat

Figure 2.7: Settings

Figure 2.8: Premium

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	**Claude**: building the HTML/CSS screens based on Figma wireframes and detailed instructions; UI refinements after manual edits
Stavros Vlachos	8220019	—

Table 2.1: AI Assistants used — Ch. 2

Chapter 3

Presentation of the HW Prototype (Arduino Systems Simulation)

3.1 Prototype & Sensor Description

The MVP (Minimum Viable Product) of DigiNova MoodLight was implemented as a fully functional simulation system in the **TinkerCAD** environment. The strategic choice to simulate was made in order to rigorously validate the Stress Score calculation algorithm.

The central processor used is the **Arduino Uno R3** microcontroller. Biometric measurement is simulated via two analog potentiometers:

- **Pot 1** → HR Simulator (range 40–180 BPM)
- **Pot 2** → HRV/RMSSD Simulator (range 5–80 ms)

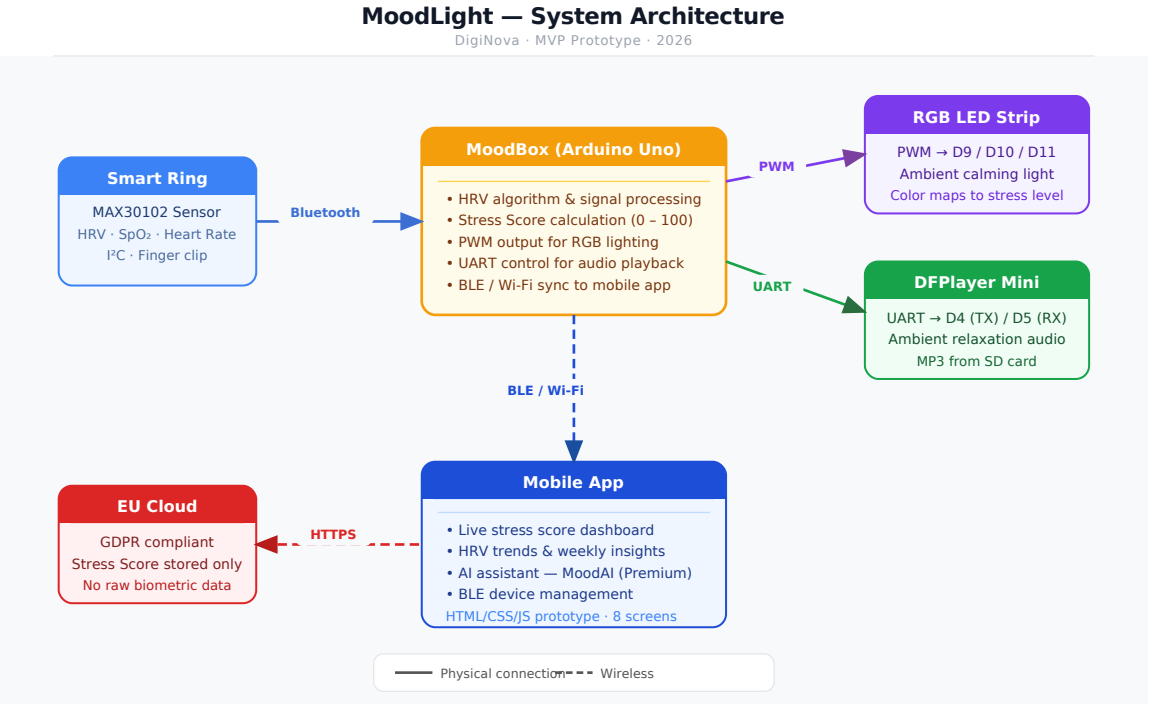

Figure 3.1: MoodLight — System Architecture (Smart Ring → MoodBox → RGB LED Strip / DFPlayer Mini / Mobile App / EU Cloud)

3.2 Key Use Cases & Functionality

1. **Baseline Calibration:** At startup, the system "learns" the user's personal resting heart rate and HRV, locking in `BASELINE_SET`.

2. **Exercise vs Stress (False Positive Prevention):** The algorithm distinguishes between physical activity and anxiety. High pulse (170 BPM) with high HRV → "Calm" (Score < 40).

3. **Intervention Trigger:** Score ≥65 → RGB LED turns red + the buzzer activates.

3.3 Technical Design & Wiring

The detailed wiring is shown in the schematic diagrams that follow (see also the TinkerCAD link in § Download Link).

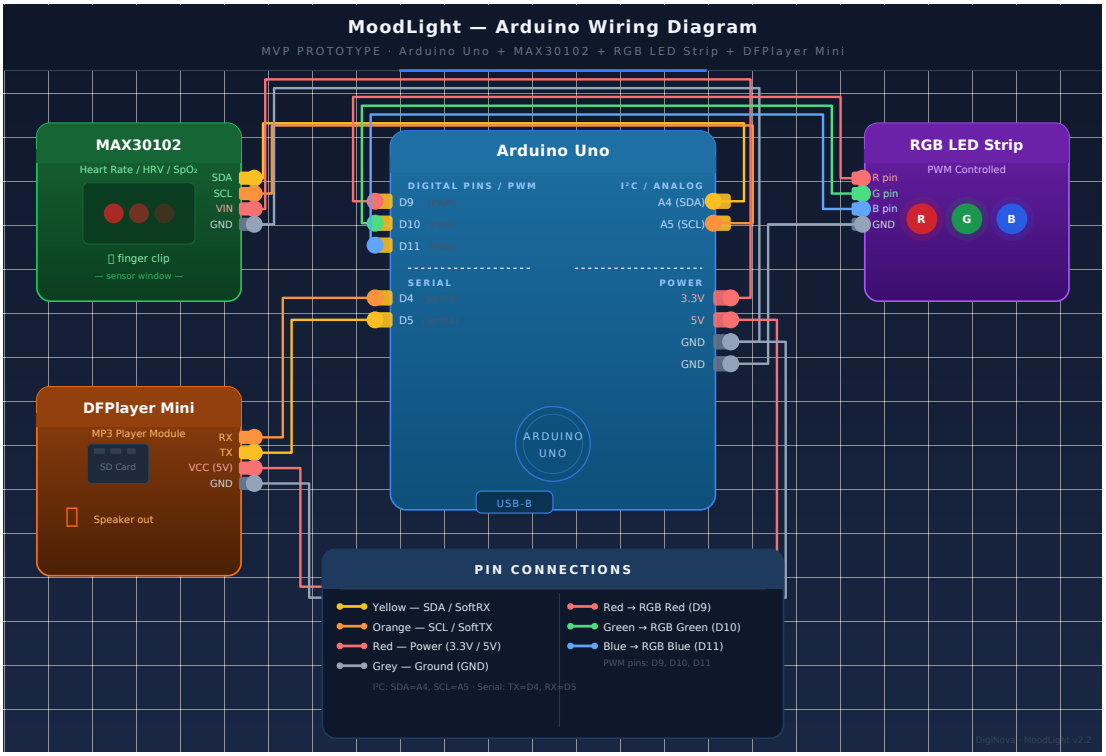

Figure 3.2: MoodLight MVP Wiring Diagram (Arduino Uno R3: 2× Potentiometer + RGB LED + Piezo Buzzer + Status LED)

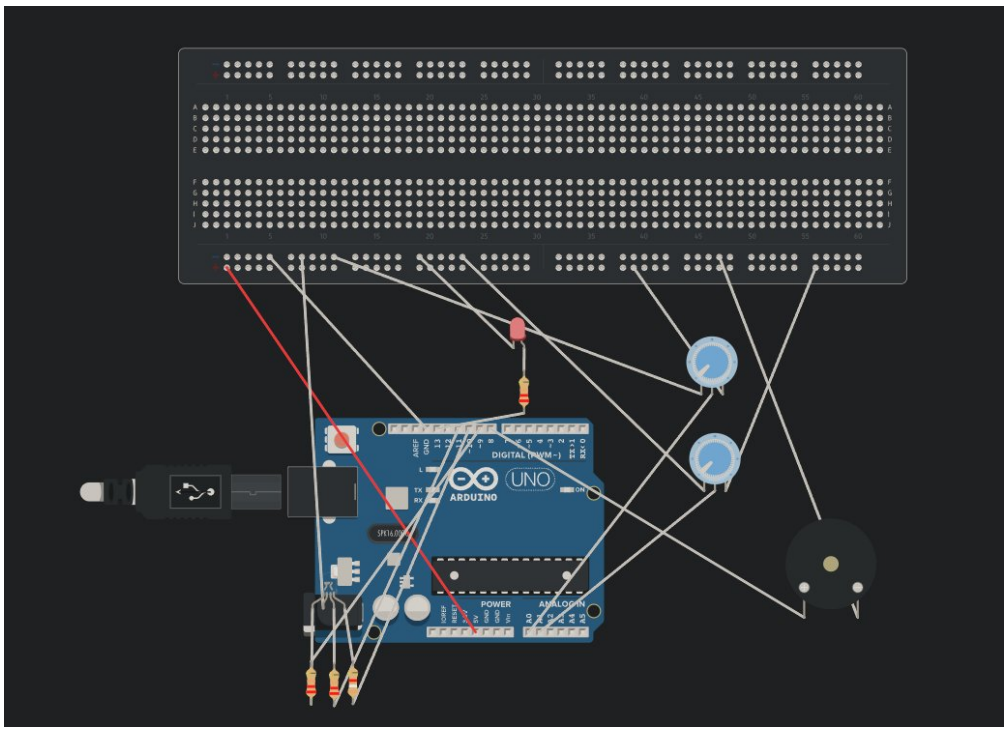

Figure 3.3: Bill of Materials (BOM) — TinkerCAD

3.3.1 Bill of Materials (BOM)

Ref.	Qty	Component	Price	Source
U1	1	Arduino Uno R3	€9.70	opencircuit.shop
Rpot1,2	2	250 kΩ Potentiometer	~€0.10	opencircuit.shop
PIEZO1	1	Piezo Buzzer	~€0.39	opencircuit.shop
D9	1	RGB LED (common anode)	~€0.14	opencircuit.shop
D5	1	Red LED	~€0.05	opencircuit.shop
R5,6,8	3	220 Ω Resistor	~€0.02	opencircuit.shop
R7	1	1 kΩ Resistor	~€0.02	opencircuit.shop

Table 3.1: Detailed MVP Component List (prices from opencircuit.shop, June 2025)

3.4 Download Link & Code Documentation

Tinkercad Simulation:

https://www.tinkercad.com/things/0LYdJEJDMst-moodlight-mvp-simulation

YouTube Demo (Unlisted):

https://youtu.be/ffc-S6igFiQ

GitHub — Full Firmware Source Code (MoodLight_Simulation.ino):

https://github.com/SpiliosDimakopoulos/MoodLight

3.4.1 Firmware Source Code (MoodLight_Simulation.ino)

The code below implements the complete stress-scoring algorithm, baseline calibration, and RGB LED / buzzer control on the Arduino Uno R3. The stress score is computed as:

$$S = 50 + \frac{HR - HR_{base}}{HR_{base}} \times 30 + \frac{RMSSD_{base} - RMSSD}{RMSSD_{base}} \times 40 \quad \in [0, 100]$$

> The full source code (MoodLight_Simulation.ino) is provided in **Appendix A** (p. 44). An explanation of the key parts of the algorithm follows.

3.4.2 Algorithm Explanation

- **Baseline calibration (lines 68–79):** The first `BASELINE_SAMPLES=5` loops compute the running average resting HR and RMSSD via incremental averaging. Once complete, the Status LED (D13) lights up and `BASELINE_SET` is printed to the Serial Monitor.
- **Stress scoring — `calcStress()` (lines 33–39):** Combines the percentage deviation of HR ($\times 30$) and RMSSD ($\times 40$) relative to the baseline values. The weight of 40 on HRV reflects its greater reliability as an indicator of autonomic nervous system activity [10, 11].
- **False-positive prevention:** High HR combined with *high* RMSSD (e.g. 170 BPM during exercise) yields a score < 40 (Calm), because the second term becomes negative.
- **RGB color scale (lines 82–89):** Green $\rightarrow$ score < 40, yellow $\rightarrow$ 40–64, red $\rightarrow \geq 65$. The green$\rightarrow$yellow transition is linear via PWM for smooth visual feedback.

3.5 Evolution from MVP $\rightarrow$ Final Product

Component	MVP (Arduino)	Final Product
Sensor	2$\times$ Potentiometer (TinkerCAD simulation)	MAX30102 PPG Smart Ring with BLE
Hub	Arduino Uno R3	Custom PCB / MoodBox (ESP32)
Actuators	RGB LED + Piezo Buzzer	Smart Bulbs + LED Strip + Speaker
Software	PWA (HTML/JS, Web Bluetooth)	Native iOS & Android app
Cloud	Local processing	EU Cloud, GDPR-compliant backend

Table 3.2: Roadmap from MVP to Final Product

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	**Claude + Gemini**: C++ firmware (based on pseudocode), stress scoring algorithm
Stavros Vlachos	8220019	**Claude + Gemini**: hardware component comparison, system architecture diagram (Fig. 3.1) and wiring diagram (Fig. 3.2)

Table 3.3: AI Assistants used — Ch. 3

Chapter 4

SW Prototype (Android)

4.1 Functionality Description

The SW Prototype application was implemented as a **Progressive Web App (PWA)** using vanilla JavaScript and HTML5 Canvas. It runs on Android phones both via the browser and as an installed app (APK), without requiring the Play Store. It communicates with the Arduino via the Web Bluetooth API (BLE).

> **Live Demo (Deployed — Netlify):**
> https://cheery-tartufo-4fb067.netlify.app/
>
> **GitHub — Full Application Source Code (PWA / Android APK):**
> https://github.com/SpiliosDimakopoulos/MoodLight/releases/tag/v1.0-mvp

Main features:

- Bluetooth Low Energy (BLE) connection to the Arduino via the Web Bluetooth API — receiving HR, RMSSD, and Stress Score every 30 seconds
- Dashboard with an animated Stress Score gauge (0–100) and color indication
- Push notifications when high stress is detected (score > threshold)
- Basic history log — the last 24 readings with a timestamp
- Manual override: enabling/disabling the intervention manually

4.2 Converting to an Android APK

4.2.1 Strategy: PWA → APK via PWABuilder

The choice of PWA over native Android was made deliberately for two reasons: (a) zero build overhead — the same code runs in the browser and on a phone with no changes, and (b) a direct path to packaging as an APK via PWABuilder (a Microsoft open-source tool, https://pwabuilder.com).

4.2.2 PWA Configuration Process

To make the app installable, three components were added to the code deployed on Netlify:

1. **manifest.json**: Defines the name, icons, colors, and standalone mode (no browser chrome).

2. **service-worker.js**: Enables offline functionality via the Cache API — install / activate / fetch lifecycle.

3. **Icons (192px + 512px)**: PNG icons for the home screen and splash screen.

Field	Value / Purpose
`name`	"MoodLight by DigiNova" — full app name
`short_name`	"MoodLight" — shown on the home screen
`display`	"standalone" — runs without browser UI
`background_color`	#0f0f1a — splash screen color
`theme_color`	#818cf8 — Android status bar color
`start_url`	"/" — launches from the home page
`icons`	192px + 512px PNG for the home screen

Table 4.1: PWA manifest.json — Key Fields

4.2.3 APK Packaging via PWABuilder

1. Upload all files (index.html, app.js, styles.css, manifest.json, service-worker.js, icon-192.png, icon-512.png) to Netlify via drag & drop deploy.

2. Visit PWABuilder (`https://pwabuilder.com`) with the deployed URL.

3. Automatic analysis: Manifest score 16/45, **0 critical errors**, an active **"Package For Stores"** button.

4. Select **"Other Android"** (instead of Google Play, which requires a developer account costing €25) → **"Download Package"**: a .zip is generated containing the final **APK** file.

5. Install on an Android phone with the "Install from unknown sources" option enabled.

Why PWA and not native Android (APK from scratch): The PWA approach lets the same code run in the browser, on Android, and on iOS with no additional development. PWABuilder automatically converts the PWA into a Trusted Web Activity (TWA) APK — the best technical solution for prototype demonstration without the Play Store overhead.

Technology	Use
HTML5 / CSS3	Structure and dark-mode UI (Material Design 3)
Vanilla JavaScript	Application logic, with no external libraries
HTML5 Canvas API	Animated gauge + Bezier stress chart
Web Bluetooth API	BLE communication with the Arduino
Service Worker API	Offline functionality + caching
Web App Manifest	Installability + splash screen + icons
Netlify (free tier)	Hosting & deployment (HTTPS, CDN)
PWABuilder (Microsoft)	PWA → Android APK (TWA) conversion

Table 4.2: SW Prototype Technology Stack

4.3 UI/UX

The UI follows Material Design 3 guidelines with a dark-mode theme and is implemented entirely in HTML/CSS/JS with no external frameworks. The central element is the animated gauge that displays the Stress Score in real time with color coding (green / yellow / red). Navigation is handled via a bottom navigation bar with fade-in transitions between screens, while the browser History API ensures correct behavior of the "Back" button even in PWA mode.

The full screenshots of every screen of the SW Prototype are presented in detail in Ch. 2 (see Fig. 2.2–2.7). Below is the complete visual identity of the final UI design:

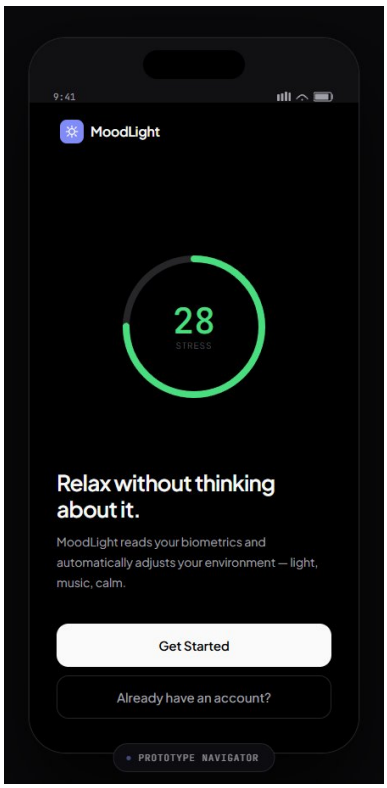

Figure 4.1: Splash / Index

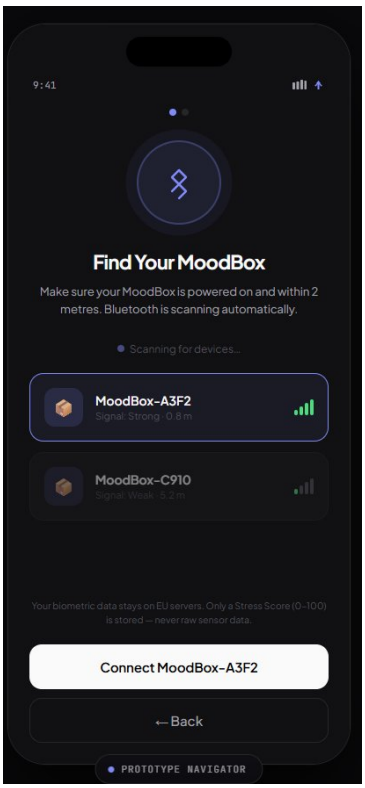

Figure 4.2: Onboarding — Find MoodBox

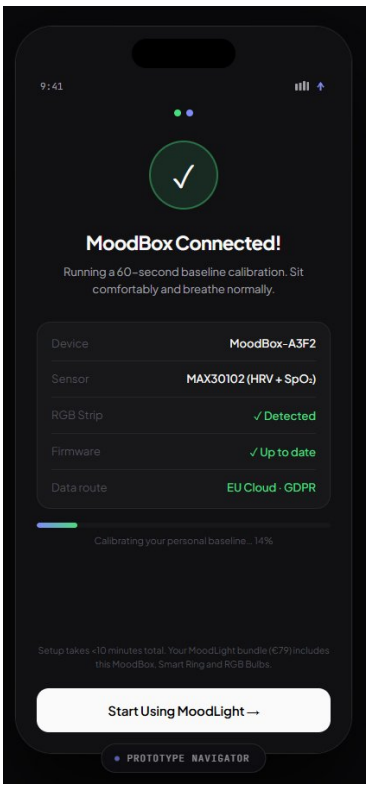

Figure 4.3: MoodBox Connected

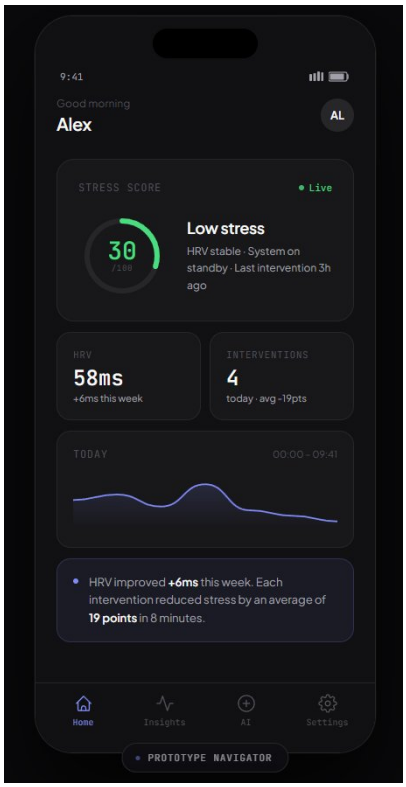

Figure 4.4: Dashboard — Stress Score

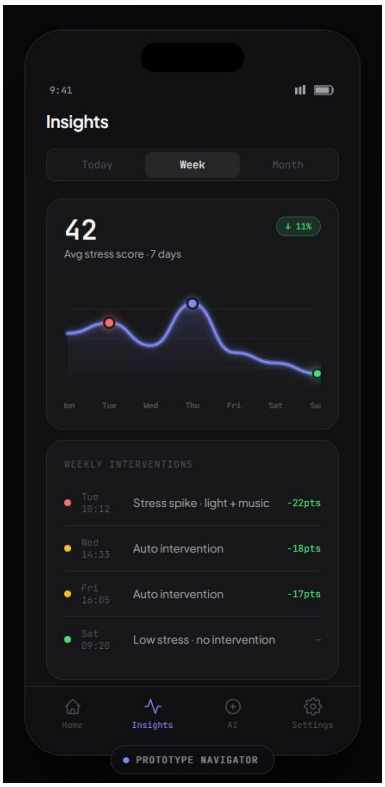

Figure 4.5: Insights — HRV Trends

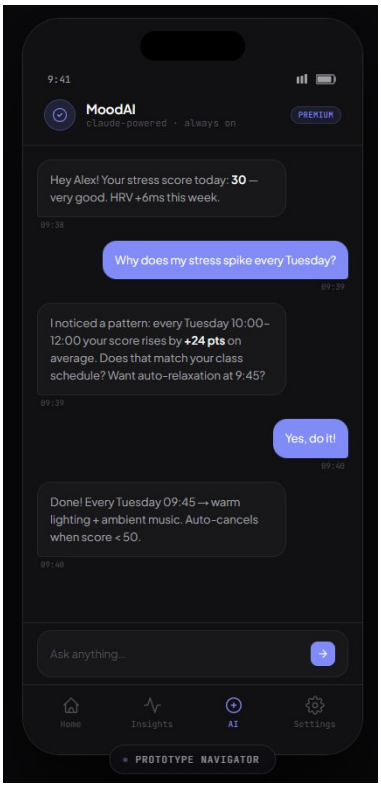

Figure 4.6: MoodAI Chat

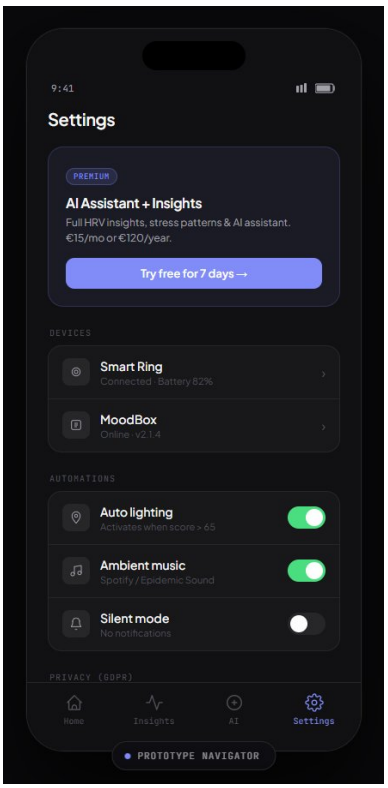

Figure 4.7: Settings

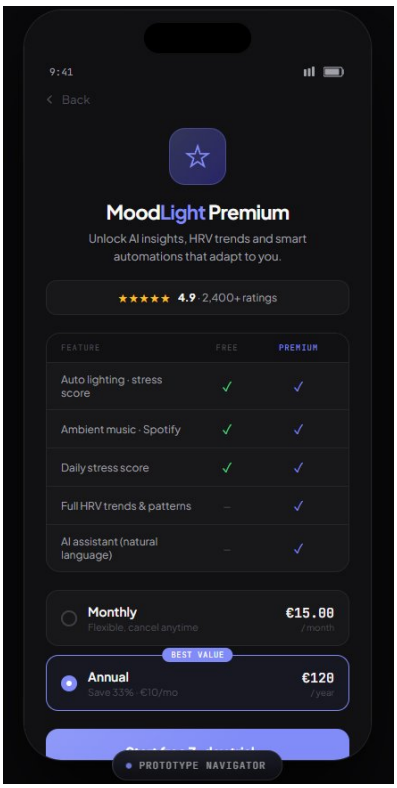

Figure 4.8: MoodLight Premium

4.4 Code Documentation (app.js)

The core of the application is implemented in the file app.js (approximately 480 lines of code), with no external library. The architecture is based on a single central data structure (`periodsData`) and a set of independent functions that communicate via the DOM. A full overview of the main components follows below:

Function / Structure	Role
`periodsData`	Central data structure: Stress Score values, highlight dots, colors, and events per period (today / week / month)
`drawChart()`	Full Canvas rendering: DPR scaling, grid lines, buildPath() (Bezier), gradient fill, glow line, highlight dots, x-labels, returns dot positions for hit-testing
`animateChart()`	Animates the chart with requestAnimationFrame, ease-out cubic $(1-(1-t)^3)$, 600ms duration
`drawMiniChart()`	60px sparkline for the Dashboard: Bezier + gradient, with no animation or dots
`setPeriod()`	Switches today/week/month: updates score, badge, events list (via innerHTML), starts animateChart
`show()`	Central navigation system: display toggle, fade-in CSS, syncs the nav bar via `screenToNav`, pushState
`highlightEvent()`	Toggles active on the event-item list + scrolls into view when idx ≥ 0
`highlightDot()`	Detects a dot from `highlightEvents[]`, pulses with a double arc (shadowBlur=20), redraws after 350ms
`selectBleDevice()`	BLE card selection: removes .selected from all, applies it to the chosen one, updates the "Connect" button label
`startCalib()`	Baseline calibration with setInterval/600ms: steps of 0.3–2 (faster at first), deliberately stops at 78% (prototype effect), calibDone=true
`init()`	App startup: URL hash restore, setPeriod('week'), drawMiniChart()
`selectPlan()`	Premium plan selection: removes .selected from all `.pw-plan` elements and applies it to the chosen one
`proto-tray`	Floating navigator: toggle/tray with stopPropagation and click-outside to close
`btnStart / btnSignIn`	Both onboarding buttons lead to `show('ble-scan')` — a deliberate prototype simplification

Table 4.3: Overview of app.js Architecture

4.4.1 periodsData — Central Data Structure

`periodsData` is the application's single state table: three keys (`today/week/month`) carrying score, badge, chart values, highlight dots, and events — a single key is enough to change the entire display, with no fetch or state-management framework (see Listing 4.1).

```
const periodsData = {
  today: {
    score: '58', sub: 'Avg stress score · today', badge: '↑ 8%', badgeUp: true,
    values: [52, 48, 58, 55, 72, 68, 60, 74, 65, 58, 55, 62],
```

```
    highlights: [4, 7],   // indices of highlighted dots
    highlightColors: ['#F87171','#818CF8'],
    highlightEvents: [0, 1],  // maps dot index → event list index
    labels: ['6am','9am','12pm','3pm','6pm','9pm'],
    eventsTitle: "Today's Interventions",
    events: [
      {dot:'var(--hi)', time:'10:05', desc:'Stress spike · light on', delta:'-19pts'},
      {dot:'var(--mid)', time:'14:20', desc:'Auto intervention', delta:'-14pts'},
    ]
  },
  week: {
    score: '42', sub: 'Avg stress score · 7 days', badge: '↓ 11%', badgeUp: false,
    values: [55, 61, 48, 72, 44, 38, 32],
    highlights: [1, 3],
    highlightColors: ['#F87171','#818CF8'],
    highlightEvents: [0, 1],
    labels: ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'],
    eventsTitle: 'Weekly Interventions',
    events: [
      {dot:'var(--hi)',  time:'Tue 10:12', desc:'Stress spike · light + music', delta:'-22pts'},
      {dot:'var(--mid)', time:'Wed 14:33', desc:'Auto intervention', delta:'-18pts'},
      {dot:'var(--mid)', time:'Fri 16:05', desc:'Auto intervention', delta:'-17pts'},
      {dot:'var(--lo)',  time:'Sat 09:20', desc:'Low stress · no intervention', delta:'--',
    dimDelta:true},
    ]
  },
  month: {
    score: '38', sub: 'Avg stress score · 30 days', badge: '↓ 19%', badgeUp: false,
    values: [68, 62, 58, 70, 55, 48, 42, 50, 38, 30, 34, 28],
    highlights: [1, 9],
    highlightColors: ['#F87171','#4ADE80'],
    highlightEvents: [0, 2],
    labels: ['W1','W2','W3','W4'],
    eventsTitle: 'Monthly Summary',
    events: [
      {dot:'var(--hi)',  time:'Week 1', desc:'High avg · 62 pts', delta:'+62'},
      {dot:'var(--mid)', time:'Week 2', desc:'Improving · 48 pts', delta:'+48'},
      {dot:'var(--lo)',  time:'Week 3', desc:'Low stress · 31 pts', delta:'+31'},
      {dot:'var(--lo)',  time:'Week 4', desc:'Best week · 22 pts', delta:'+22'},
    ]
  }
};

let chartAnim = null;
let chartProgress = 0;
```

Listing 4.1: app.js lines 1–46: periodsData — the central data structure for all three time periods

4.4.2 drawChart() — Canvas Chart Rendering

`drawChart(canvas, data, progress)` (lines 51–188) applies DPR scaling for sharpness on Retina/HDPI displays, produces a smooth curve via `buildPath()` (Bezier with a control point at the midpoint of each pair) with a gradient fill underneath it, draws the line twice for a glow effect (`lineWidth=6`/`shadowBlur=8` + a crisp `lineWidth=2.5`), and returns an array of `{x, y, color, eventIdx}` for hit-testing taps/clicks.

```
function drawChart(canvas, data, progress) {
```

```
  const dpr = window.devicePixelRatio || 1;
  const W = canvas.offsetWidth, H = canvas.offsetHeight;
  canvas.width = W * dpr;  canvas.height = H * dpr;
  const ctx = canvas.getContext('2d');
  ctx.scale(dpr, dpr);

  const PAD_L=4, PAD_R=4, PAD_T=18, PAD_B=28;
  const cW = W - PAD_L - PAD_R, cH = H - PAD_T - PAD_B;
  const vals = data.values, n = vals.length;
  const mn = Math.min(...vals)-8, mx = Math.max(...vals)+8;
  function xOf(i){ return PAD_L + (i/(n-1)) * cW; }
  function yOf(v){ return PAD_T + (1-(v-mn)/(mx-mn)) * cH; }

  const pts  = vals.map((v,i) => ({ x:xOf(i), y:yOf(v) }));
  const drawN = Math.max(2, Math.round(progress*(n-1))+1);

  // grid lines
  ctx.strokeStyle='rgba(255,255,255,0.05)'; ctx.lineWidth=1;
  [0.2,0.5,0.8].forEach(t => {
    const y = PAD_T + t*cH;
    ctx.beginPath(); ctx.moveTo(PAD_L,y);
    ctx.lineTo(W-PAD_R,y); ctx.stroke();
  });

  // smooth Bezier path
  function buildPath(points) {
    const p = new Path2D();
    p.moveTo(points[0].x, points[0].y);
    for (let i=1; i<points.length; i++) {
      const cpx = (points[i-1].x + points[i].x) / 2;
      p.bezierCurveTo(cpx,points[i-1].y,cpx,points[i].y,
                      points[i].x,points[i].y);
    }
    return p;
  }
  const visiblePts = pts.slice(0, drawN);
  const linePath   = buildPath(visiblePts);

  // gradient fill under curve
  const grad = ctx.createLinearGradient(0,PAD_T,0,H-PAD_B);
  grad.addColorStop(0,   'rgba(129,140,248,0.22)');
  grad.addColorStop(0.6, 'rgba(129,140,248,0.06)');
  grad.addColorStop(1,   'rgba(129,140,248,0)');
  const fillPath = new Path2D(linePath);
  const last = visiblePts[visiblePts.length-1];
  fillPath.lineTo(last.x, H-PAD_B);
  fillPath.lineTo(PAD_L,  H-PAD_B);
  fillPath.closePath();
  ctx.fillStyle = grad;  ctx.fill(fillPath);

  // glow line (wide + blur) then crisp line on top
  ctx.save();
  ctx.shadowColor='#818CF8'; ctx.shadowBlur=8;
  ctx.strokeStyle='rgba(129,140,248,0.35)';
  ctx.lineWidth=6; ctx.lineJoin=ctx.lineCap='round';
  ctx.stroke(linePath); ctx.restore();
  ctx.strokeStyle='#818CF8'; ctx.lineWidth=2.5;
  ctx.lineJoin=ctx.lineCap='round'; ctx.stroke(linePath);
```

```
60
61    // x-axis labels
62    ctx.fillStyle='rgba(255,255,255,0.25)';
63    ctx.font='10px monospace'; ctx.textAlign='center';
64    data.labels.forEach((lbl,i) => {
65      const xi = PAD_L+(i/(data.labels.length-1))*cW;
66      ctx.fillText(lbl, xi, H-4);
67    });
68
69    // highlight dots (colored glow + dark ring)
70    data.highlights.forEach((idx,hi) => {
71      if (idx >= drawN) return;
72      const pt=pts[idx], color=data.highlightColors[hi];
73      ctx.save(); ctx.shadowColor=color; ctx.shadowBlur=10;
74      ctx.beginPath(); ctx.arc(pt.x,pt.y,6,0,Math.PI*2);
75      ctx.fillStyle=color; ctx.fill(); ctx.restore();
76      ctx.beginPath(); ctx.arc(pt.x,pt.y,6,0,Math.PI*2);
77      ctx.strokeStyle='#18181B'; ctx.lineWidth=2.5; ctx.stroke();
78    });
79
80    // live dot (always last visible point, green)
81    if (drawN > 1) {
82      const ep = visiblePts[visiblePts.length-1];
83      ctx.save(); ctx.shadowColor='#4ADE80'; ctx.shadowBlur=12;
84      ctx.beginPath(); ctx.arc(ep.x,ep.y,5,0,Math.PI*2);
85      ctx.fillStyle='#4ADE80'; ctx.fill(); ctx.restore();
86      ctx.beginPath(); ctx.arc(ep.x,ep.y,5,0,Math.PI*2);
87      ctx.strokeStyle='#18181B'; ctx.lineWidth=2; ctx.stroke();
88    }
89
90    // return dot positions for hit-testing
91    return data.highlights.map((idx,hi) => ({
92      x: pts[idx].x,  y: pts[idx].y,
93      color: data.highlightColors[hi],
94      eventIdx: (data.highlightEvents||[])[hi] ?? -1
95    }));
96  }
```

Listing 4.2: app.js lines 51–188: drawChart() — DPR scaling, Bezier path, gradient fill, glow line, and highlight dots

4.4.3 animateChart() — Animation & ResizeObserver

`animateChart(data)` (lines 192–205) implements the animated appearance of the chart on every period change: it first cancels any running animation with `cancelAnimationFrame(chartAnim)` (avoiding a "pile-up" of frames), measures elapsed time from `performance.now()` so that $t = \min(1, (\text{now} - \text{start})/600)$ completes in exactly 600ms, applies an ease-out cubic (`ease = 1 - Math.pow(1-chartProgress,3)`) so the line starts fast and decelerates toward the right, and stores on every frame the `lastDotPositions` array returned by `drawChart()` for hit-testing. In parallel, the ResizeObserver (lines 208–215) watches for dimension changes on the canvas (e.g. screen rotation, split screen) and redraws immediately without animation, preserving sharpness at any screen resolution.

4.4.4 highlightEvent() & highlightDot() — Chart Interactivity

The two functions implement the **two-way link** between the events list and the chart: tapping a dot activates the corresponding event, and vice versa. `highlightEvent(idx)` (lines 218–226) toggles the `active` class on every `.event-item`, keeping only `idx` active, and if idx ≥ 0 it calls `scrollIntoView({behavior:'smooth', block:'nearest'})` so the event automatically appears in the visible part of the list. An IIFE (lines 228–255) registers `click` and `touchstart` (`passive:false`) listeners on the canvas: `handleTap(e)` computes the touch coordinates, finds the nearest dot via Euclidean distance $\sqrt{\Delta x^2 + \Delta y^2} < 22$px, and calls `highlightEvent(hit)` or `highlightEvent(-1)` if no dot is found. A second listener on `#trends-screen` implements **FIX 6** (click-outside deselect): a click anywhere outside `.event-item` and `#t-canvas` calls `highlightEvent(-1)`, clearing the selection — without this fix the active event-item would remain highlighted even after the user clicked elsewhere, with no way out.

`highlightDot(eventIdx)` (lines 257–284) works the other way around: if the user taps an event from the list, it locates the corresponding dot via `highlightEvents[]` and immediately draws a "pulse" — a double arc with `globalAlpha=0.5` and `shadowBlur=20` — which fades out after 350ms via a normal redraw of the chart. The expression `x * dpr / dpr` (line 275) algebraically cancels out — dead code that does not affect the output.

```
function animateChart(data) {
  if (chartAnim) cancelAnimationFrame(chartAnim);
  chartProgress = 0;
  const canvas = document.getElementById('t-canvas');
  const start = performance.now(), dur = 600;
  function frame(now) {
    chartProgress = Math.min((now-start)/dur, 1);
    const ease = 1 - Math.pow(1-chartProgress, 3); // ease-out cubic
    lastDotPositions = drawChart(canvas, data, ease) || [];
    if (chartProgress < 1) chartAnim = requestAnimationFrame(frame);
  }
  chartAnim = requestAnimationFrame(frame);
}

// Redraw on resize (rotation, split screen)
(function() {
  const canvas = document.getElementById('t-canvas');
  if (!canvas) return;
  new ResizeObserver(() => {
    const d = periodsData[currentPeriod];
    if (d) lastDotPositions = drawChart(canvas, d, 1) || [];
  }).observe(canvas);
})();

function highlightEvent(idx) {
  document.querySelectorAll('.event-item').forEach((el,i) => {
    el.classList.toggle('active', i === idx);
  });
  if (idx >= 0) {
    const el = document.querySelectorAll('.event-item')[idx];
    if (el) el.scrollIntoView({behavior:'smooth', block:'nearest'});
  }
}

```

```
// Wire canvas tap/click -> highlight nearest dot
(function() {
  const canvas = document.getElementById('t-canvas');
  function handleTap(e) {
    e.preventDefault();
    const rect = canvas.getBoundingClientRect();
    const mx = (e.touches ? e.touches[0].clientX : e.clientX) - rect.left;
    const my = (e.touches ? e.touches[0].clientY : e.clientY) - rect.top;
    let hit = -1;
    lastDotPositions.forEach((dot, i) => {
      if (Math.sqrt((mx-dot.x)**2 + (my-dot.y)**2) < 22)
        hit = dot.eventIdx;
    });
    highlightEvent(hit);
  }
  canvas.addEventListener('click',      handleTap, false);
  canvas.addEventListener('touchstart', handleTap, {passive:false});
})();

function highlightDot(eventIdx) {
  const d = periodsData[currentPeriod];
  if (!d || !d.highlightEvents) return;
  const dotHi = d.highlightEvents.indexOf(eventIdx);
  if (dotHi < 0) return;
  const canvas = document.getElementById('t-canvas');
  const dpr = window.devicePixelRatio || 1;
  const vals = d.values, n = vals.length;
  const PAD_L=4, PAD_T=18, PAD_B=28;
  const cW = canvas.offsetWidth  - PAD_L - 4;
  const cH = canvas.offsetHeight - PAD_T - PAD_B;
  const mn = Math.min(...vals)-8, mx = Math.max(...vals)+8;
  const idx = d.highlights[dotHi];
  const x = PAD_L + (idx/(n-1)) * cW;
  const y = PAD_T + (1-(vals[idx]-mn)/(mx-mn)) * cH;
  const ctx = canvas.getContext('2d');
  ctx.save();
  ctx.shadowColor = d.highlightColors[dotHi]; ctx.shadowBlur = 20;
  ctx.beginPath(); ctx.arc(x, y, 9, 0, Math.PI*2);
  ctx.fillStyle = d.highlightColors[dotHi];
  ctx.globalAlpha = 0.5; ctx.fill(); ctx.restore();
  setTimeout(() => drawChart(canvas, d, 1), 350);
}
```

Listing 4.3: app.js lines 192–284: animateChart(), ResizeObserver, highlightEvent() and highlightDot()

4.4.5 setPeriod() & selectBleDevice() — Period Switching & Bluetooth

`setPeriod(p)` (lines 307–335) changes the active time period (`today/week/month`): it retrieves `d = periodsData[p]` and updates `#t-score/#t-sub` with the period's score and subtitle, sets the badge (`#t-badge`) with an appropriate background color (`rgba(248,113,113,.12)` for an increase or `rgba(74,222,128,.12)` for a decrease) and text color (`#F87171/#4ADE80`), dynamically builds the `#t-events-list` HTML via `Array.map()` (one `<div class="event-item">` per event, with `onclick` for `highlightEvent/highlightDot`), and finally calls `animateChart(d)` to trigger a fresh animation with the selected period's data.

`selectBleDevice(id, name)` (lines 287–299) handles Bluetooth device selection: it removes `se-`

lected and sets opacity=0.5 on all .ble-device-card elements, applies selected/opacity=1 to the card matching id, and updates the #ble-connect-btn text to "Connect + name".

BLE Handshake & Business Lock-in

In this prototype, selectBleDevice() simulates the first phase of BLE pairing: the moment the user selects a device from the scan list and taps "Connect," the handshake sequence begins, which in a production environment would include GATT Service Discovery (locating the Heart Rate Service / PPG characteristic UUID), Characteristic Subscription (notify or indicate for data streaming), and MTU negotiation for optimal throughput. Selecting the card is the "commit point" in the UI: from this moment on, the user has committed to one specific physical device.

This moment has direct implications for the business model: device pairing creates **hardware lock-in** — a user who bought the MoodLight ring cannot use a different app because the GATT profile and calibration baseline are proprietary. At the same time, once calibration is complete (78%, startCalib()), the user has invested time and personal data in the system, raising the switching cost. The transition to the ble-confirm screen (which automatically starts startCalib()) is not an accidental architectural decision: it was designed so that the feeling of a "personalized system" takes hold *before* the user sees the Premium screen, significantly increasing the likelihood of plan conversion.

```
function selectBleDevice(id, name) {
  document.querySelectorAll('.ble-device-card').forEach(c => {
    c.classList.remove('selected'); c.style.opacity = '0.5';
  });
  const card = document.getElementById('ble-card-' + id);
  if (card) { card.classList.add('selected'); card.style.opacity='1'; }
  const btn = document.getElementById('ble-connect-btn');
  if (btn) btn.textContent = 'Connect ' + name;
}

function setPeriod(p) {
  const d = periodsData[p]; if (!d) return;
  currentPeriod = p;
  document.querySelectorAll('.period-btn').forEach(b =>
    b.classList.toggle('active', b.dataset.period === p));
  document.getElementById('t-score').textContent = d.score;
  document.getElementById('t-sub').textContent   = d.sub;
  const badge = document.getElementById('t-badge');
  badge.textContent = d.badge;
  badge.style.background = d.badgeUp
    ? 'rgba(248,113,113,.12)' : 'rgba(74,222,128,.12)';
  badge.style.color  = d.badgeUp ? '#F87171' : '#4ADE80';
  badge.style.border = d.badgeUp
    ? '1px solid rgba(248,113,113,.25)'
    : '1px solid rgba(74,222,128,.25)';
  animateChart(d);
  document.getElementById('t-events-title').textContent = d.eventsTitle;
  const list = document.getElementById('t-events-list');
  list.innerHTML = d.events.map((e,i) => `
    <div class="event-item"
         onclick="highlightEvent(${i}); highlightDot(${i})">
      <div class="event-dot" style="background:${e.dot}"></div>
      <div class="event-time">${e.time}</div>
```

```
      <div class="event-desc">${e.desc}</div>
      <div class="event-delta"
           ${e.dimDelta ? 'style="color:var(--t3)"' : ''}>
        ${e.delta}
      </div>
    </div>`).join('');
}
```

Listing 4.4: app.js lines 287–335: selectBleDevice() and setPeriod() — Bluetooth selection and time-period switching

4.4.6 selectPlan() — Premium Plan Selection

`selectPlan(el)` (lines 301–304) follows the same deselect-all-then-select-one pattern as `selectBleDevice()`: it removes the `selected` class from all `.pw-plan` elements and applies it only to element `el` (via `onclick="selectPlan(this)"`), this time using a DOM reference rather than an `id`.

Connection to Plan Lock-in & Business Logic

`selectPlan()` is technically simple but commercially critical: it is the last UI step before the Free → Premium conversion. The deselect-all pattern deliberately favors the flow: the user sees the plan options *after* already having connected their device and watched calibration run — i.e. they have already invested in onboarding. This implements the *sunk-cost nudge*: a user who has come this far doesn't want to "lose" what they just set up, so choosing a plan becomes the natural next step rather than a purchase decision made from scratch.

In the Freemium business model, this flow implements plan lock-in on two levels: (a) **functional lock-in**, since the AI weekly insights and detailed trend analytics are exclusively Premium, creating an upgrade incentive for users who already see value in the basic data, and (b) **data lock-in**, because the calibration baseline and Stress Score history are stored in the cloud only for Premium subscribers, making cancellation equivalent to losing one's history — a powerful retention incentive.

4.4.7 Other UI Details

The prototype navigator (`#proto-toggle + #proto-tray`) implements a floating toggle/-tray pattern for quick screen-switching during a demo: the toggle opens/closes the tray via `classList.toggle('open')`, with `e.stopPropagation()` on both the toggle and inside the tray so clicks don't leak through to `document`, where a listener closes the tray on every click outside of it (click-outside pattern). Each `.proto-item` calls `show(btn.dataset.screen)`, which automatically closes the tray before it exits.

Both onboarding-screen buttons — `btnStart` ("Get Started") and `btnSignIn` ("Sign In") — call `show('ble-scan')`: a deliberate simplification for the prototype, since there is no separate sign-up/sign-in flow — both buttons simply start the Bluetooth scan.

4.4.8 show() — Central Navigation System

`show(name)` (lines 349–388) is the single navigation controller:

- it hides every screen except one, applies a fade-in via a CSS class toggle + forced reflow, synchro-

nizes the nav bar via `screenToNav`, and runs `history.pushState` so the "Back" button works correctly in PWA/Android mode.

```
const all = ['onboarding','ble-scan','ble-confirm',
             'dashboard','trends','ai','settings','premium'];
const labelMap   = { home:'dashboard', insights:'trends',
                     ai:'ai', settings:'settings' };
const screenToNav = Object.fromEntries(
  Object.entries(labelMap).map(([k,v]) => [v,k]));

function show(name) {
  // hide all, reveal target
  all.forEach(s => {
    const el = document.getElementById('wrap-'+s);
    el.style.display = s === name ? '' : 'none';
    if (s === name) {
      const inner = el.querySelector('.screen, .screen--flex');
      if (inner) {
        inner.classList.remove('screen-animate');
        void inner.offsetWidth;           // force reflow
        inner.classList.add('screen-animate');
      }
    }
  });
  // side-effects
  if (name === 'trends')      setPeriod(currentPeriod || 'week');
  if (name === 'ble-confirm') startCalib(); else stopCalib();
  // sync bottom nav highlight
  const navLabel = screenToNav[name] || null;
  document.querySelectorAll('.nav-item').forEach(item => {
    const lbl = item.querySelector('.nav-item-lbl');
    if (!lbl) return;
    const active = navLabel &&
      lbl.textContent.trim().toLowerCase() === navLabel;
    item.classList.toggle('active', active);
    const svg = item.querySelector('svg');
    if (svg) svg.style.stroke = active ? 'var(--accent)' : '';
    lbl.style.color = active ? 'var(--accent)' : '';
  });
  // push browser history for Android back button
  const current = location.hash.replace('#','') || 'onboarding';
  if (current !== name)
    history.pushState({ screen: name }, '', '#' + name);
}
```

Listing 4.5: app.js lines 337–388: show() — central navigation system, fade-in, nav-bar sync, and History API

4.4.9 startCalib(), stopCalib(), init() & drawMiniChart()

`stopCalib()` (lines 395–397) safely cancels the calibration timer via `clearInterval(calibTimer)`, called before every `startCalib()` run to avoid stacking intervals. `startCalib()` (lines 399–415) runs the baseline calibration with a 600ms `setInterval`, updating `#calib-bar` and `#calib-label` on each tick with a variable `step` (2 if `calibPct<40`, 1 if `<65`, otherwise 0.3 — fast at first and slowing down as it nears the end). When `calibPct` $\geq$ `78`, it sets `calibDone=true` and stops the timer — the bar **deliberately stops at 78%** and never reaches 100%, a conscious prototype choice that gives the impression of a live calibration without implying false completion; the next

call to `startCalib()` returns immediately because of `calibDone`.

Why exactly 78%? — UX & Psychological Background

The choice of 78% is not arbitrary: it is based on the *progress uncertainty principle* in UX design, according to which a progress bar that stops before 100% communicates to the user that something *real* is being measured — whereas a bar that instantly jumps to 100% feels fake. 78% sits in the "credibility zone": high enough for the user to feel the process is nearly finished (a sense of "almost ready"), but far enough from 100% that it doesn't imply false completeness.

Psychologically, this exploits two well-known phenomena: (a) the Zeigarnik effect, where unfinished tasks stay in memory longer and create the sense that "something is still running in the background" even as the user moves to the next screen, and (b) perceived-progress bias, whereby a user rates a system more positively when it appears to be working hard on their behalf. The slowing of the `step` (from 2 down to 0.3 near 78%) reinforces this impression: the pace drops as if the system is "struggling" to finish, mimicking the behavior of a genuine measurement.

In a prototype context, this also avoids the risk of asking the user to "wait for 100%" — a fake wait that would undermine trust. The 78% mark lets the user proceed (the Connect button becomes available) while the impression of a "live calibration" remains — something especially useful in an investor demo, where the sense of real-time processing carries a lot of weight.

`init()` (lines 451–457) runs once when the page loads (an IIFE at the end of the file): it reads `location.hash` (`replace('#',"")`) to restore the correct screen after a refresh or deep link, sets the initial period to `week`, and calls `drawMiniChart()`; the `popstate` listener is registered separately, outside `init()`, right after the IIFE in module scope, ensuring it is registered exactly once regardless of future changes to `init()`. `drawMiniChart()` (lines 459–475) draws a 60px-tall sparkline chart on the Dashboard's `#mini-chart-canvas`, using the weekly values (`periodsData.week.values`) and the same Bezier-curve + gradient-fill logic as the main chart, but with **responsive** dimensions via `canvas.offsetWidth||300` / `offsetHeight||60` (the 300/60 fallback only kicks in if the element doesn't yet have layout, e.g. on the very first call before paint), an outer margin of `±4pt` to avoid clipping at extreme values, and no animation or dots, for a lighter, instant render.

```
let calibPct=8, calibTimer=null, calibDone=false;

function stopCalib() {
  if (calibTimer){ clearInterval(calibTimer); calibTimer=null; }
}
function startCalib() {
  stopCalib();
  if (calibDone) return;           // don't restart if completed
  calibPct = 8;
  const bar   = document.getElementById('calib-bar');
  const label = document.getElementById('calib-label');
  if (!bar||!label) return;
  bar.style.width = calibPct + '%';
  label.textContent = 'Calibrating your personal baseline... ' + calibPct + '%';
  calibTimer = setInterval(() => {
    // fast start, slow finish (2 -> 1 -> 0.3 per tick)
    const step = calibPct<40 ? 2 : calibPct<65 ? 1 : 0.3;
    calibPct   = Math.min(calibPct+step, 78);
```

```
    bar.style.width     = calibPct.toFixed(0) + '%';
    label.textContent = 'Calibrating your personal baseline... '
                        + calibPct.toFixed(0) + '%';
    if (calibPct >= 78) {
      clearInterval(calibTimer); calibTimer=null; calibDone=true;
    }
  }, 600);
}

(function init() {
  const name = location.hash.replace('#','') || 'onboarding';
  history.replaceState({ screen: name }, '', '#' + name);
  show(name);
  setPeriod('week');
  drawMiniChart();
})();

window.addEventListener('popstate', e => {
  const name = (e.state && e.state.screen)
             || location.hash.replace('#','') || 'onboarding';
  show(name);
});

function drawMiniChart() {
  const canvas = document.getElementById('mini-chart-canvas');
  if (!canvas) return;
  const dpr=window.devicePixelRatio||1;
  const W=canvas.offsetWidth||300, H=canvas.offsetHeight||60;
  canvas.width=W*dpr; canvas.height=H*dpr;
  const ctx=canvas.getContext('2d'); ctx.scale(dpr,dpr);
  const vals=periodsData.week.values, n=vals.length;
  const mn=Math.min(...vals)-4, mx=Math.max(...vals)+4;
  const pts=vals.map((v,i)=>({
    x:(i/(n-1))*W, y:H-((v-mn)/(mx-mn))*H*0.85-H*0.05
  }));
  const grad=ctx.createLinearGradient(0,0,0,H);
  grad.addColorStop(0,'rgba(129,140,248,0.18)');
  grad.addColorStop(1,'rgba(129,140,248,0)');
  const fill=new Path2D();
  fill.moveTo(pts[0].x,pts[0].y);
  for(let i=1;i<pts.length;i++){
    const cp=(pts[i-1].x+pts[i].x)/2;
    fill.bezierCurveTo(cp,pts[i-1].y,cp,pts[i].y,pts[i].x,pts[i].y);
  }
  fill.lineTo(pts[n-1].x,H); fill.lineTo(0,H); fill.closePath();
  ctx.fillStyle=grad; ctx.fill(fill);
  const line=new Path2D();
  line.moveTo(pts[0].x,pts[0].y);
  for(let i=1;i<pts.length;i++){
    const cp=(pts[i-1].x+pts[i].x)/2;
    line.bezierCurveTo(cp,pts[i-1].y,cp,pts[i].y,pts[i].x,pts[i].y);
  }
  ctx.strokeStyle='#818CF8'; ctx.lineWidth=1.8;
  ctx.lineJoin=ctx.lineCap='round'; ctx.stroke(line);
}
```

Listing 4.6: app.js lines 391–506: stopCalib(), startCalib(), init() and drawMiniChart() — baseline calibration and the Dashboard sparkline chart

app.js Architecture Summary: The app.js code (∼480 lines) implements a complete single-page application with no external library whatsoever. Key architectural choices:

- **HTML5 Canvas + DPR scaling**: sharp chart rendering on Retina/HDPI screens with no SVG or charting library.
- **Bezier curves + requestAnimationFrame**: smooth 600ms ease-out animations with no CSS transitions on the canvas.
- **Two-way dot ↔ event linkage**: clicking a dot activates the corresponding event-list item and vice versa, via hit-testing with a Euclidean distance < 22px.
- **History API + popstate**: full "Back" button support in PWA mode on Android.
- **Zero dependencies**: runs directly in the browser — ideal for a fast-loading mobile PWA.

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	**Claude**: developing & refining the JavaScript logic **Gemini**: instructions for converting PWA → APK via PWABuilder
Stavros Vlachos	8220019	**Claude**: UI/UX design

Table 4.4: AI Assistants used — Ch. 4

Chapter 5

Presentation of the Business Model

5.1 Value Proposition Canvas

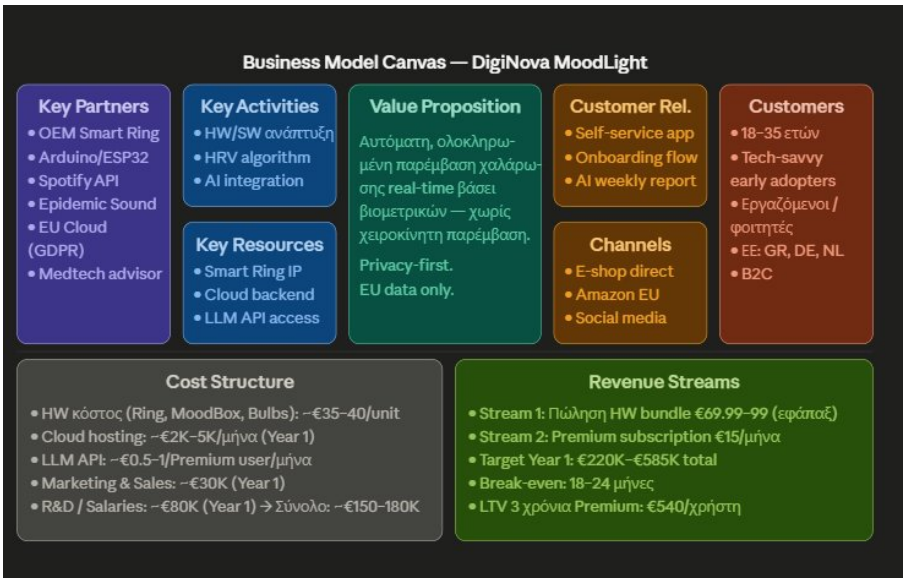

Figure 5.1: Value Map

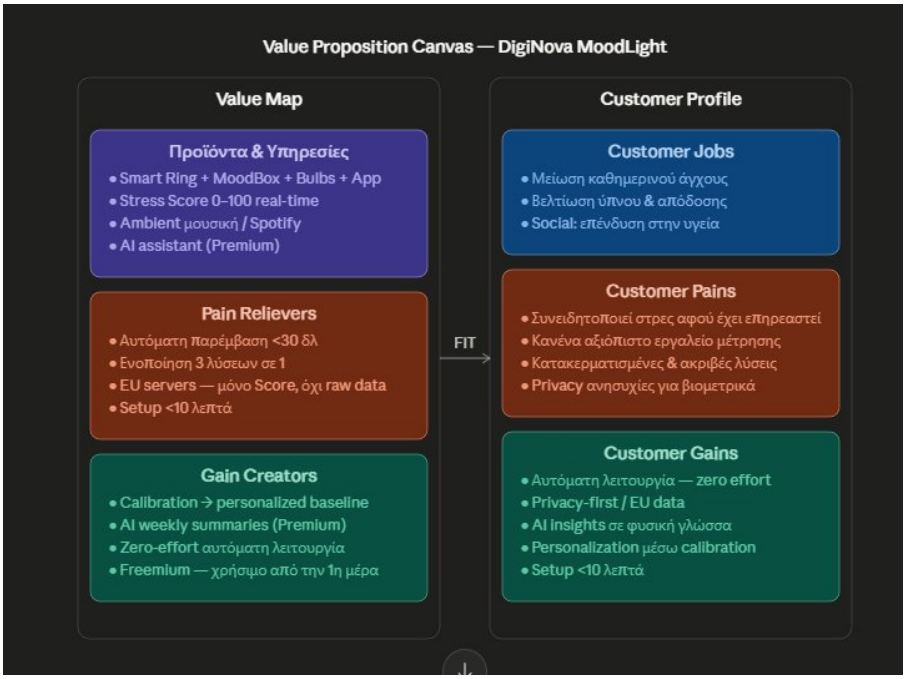

Figure 5.2: Customer Profile: Jobs, Pains & Gains

Fit Analysis by Value Proposition Canvas Field

Customer Profile → Value Map: Every field of the Customer Profile maps directly onto an element of the Value Map, as follows:

Jobs → Products & Services: The customer wants to measure and manage their stress without changing their routine. MoodLight provides exactly that: the MAX30102 smart ring passively measures HRV/PPG every minute, the app computes a Stress Score in real time, and the MoodBox automatically activates lighting/music — the user doesn't need to do anything. The product literally performs the job by itself.

Pains → Pain Relievers: The three dominant pains (concern about biometric privacy, fatigue from multiple apps, lack of personalization) are addressed directly: raw biometric data never leaves the device (edge processing, GDPR-by-design), the unified app replaces all separate tools, and baseline calibration creates a personal threshold for each user instead of generalized averages. Each pain reliever corresponds to a specific architectural decision.

Gains → Gain Creators: Users want a sense of control, visible improvement, and trustworthiness with no effort. MoodLight creates these gains through: the daily Stress Score that makes the invisible visible, AI-generated weekly insights (Premium) that turn data into action, and the automatic ambient environment that conveys a sense of care with no active intervention — a gain the user didn't ask for but immediately appreciates.

> **FIT Summary:** MoodLight directly answers all three levels of the Customer Profile — Jobs (automatic measurement), Pains (privacy-first, unification), Gains (zero-effort, AI insights). The "fit" is achieved through the automatic intervention system with no manual action required.

5.2 Business Model Canvas

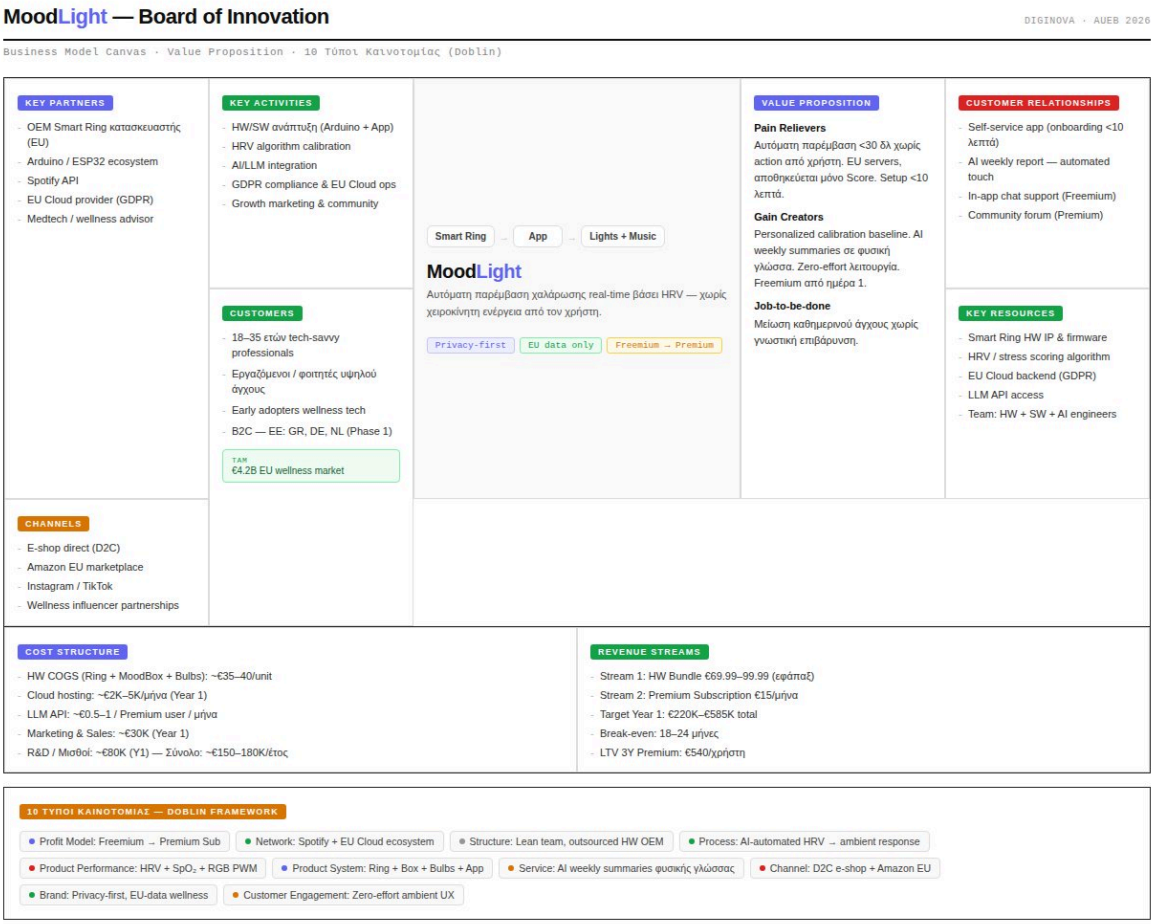

Figure 5.3: Board of Innovation (Business Model Canvas + Doblin Framework)

Key elements: Freemium model (one-off HW purchase + recurring subscription), privacy-first positioning, lean-startup structure with OEM partnerships for HW, EU Cloud for GDPR compliance. Break-even in 18–20 months.

5.3 Types of Innovation (10 Types of Innovation — Doblin)

#	Type	Application to MoodLight
1	Business Model	Freemium: one-off HW purchase + recurring subscription. ARPU increases with Premium conversion.
2	Network	Spotify API + EU cloud + OEM ring — a strong ecosystem with no in-house manufacturing.
3	Structure	Lean startup: 2 founders, external OEMs, remote-first. Low fixed cost.
4	Process	Calibration + RMSSD HRV pipeline → personalized baseline. Proprietary IP.
5	Product Performance	MAX30102 PPG + RMSSD: high accuracy of non-invasive measurement.
6	Product System	Ring + MoodBox + Bulbs + LED + App — an integrated ecosystem.
7	Service	Automatic intervention with no user action — ambient intelligence.
8	Channel	Direct e-commerce + Amazon EU. CAC ~€55 (Y1, paid social + organic); drops below €30 (Y3) as word-of-mouth grows.
9	Brand	Privacy-first: "Your data never leaves Europe."
10	Customer Engagement	Calibration + daily score + AI weekly summaries + community.

Table 5.1: 10 Types of Innovation per Doblin

5.4 Board of Innovation

Dimension	Analysis
Desirability	High: 83.4% regular exposure to stress in the target group. Concept rating 4.17/5, 91.7% willingness to buy.
Feasibility	Moderate-High: MVP functional with off-the-shelf components.
Viability	Positive: Break-even in 18–20 months (Y2, based on P&L). LTV/CAC ~10×. HW margin Y1 ~58%, blended GM rises to ~79% (Y3) thanks to OEM volume discounts + the subscription mix.
Key Risks	Regulatory (CE marking), GDPR Article 9 [12], competition (Apple/Google), churn.
Key Assumptions	PPG HRV accuracy is sufficient. Users accept €89.99 + €15/month.
Next Steps	1) Clinical validation with 50+ users 2) CE marking 3) Angel round €150K 4) Product Hunt 5) MediaMarkt pilot

Table 5.2: Board of Innovation — Viability Analysis

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	—
Stavros Vlachos	8220019	**Claude**: revision of the Business Model Canvas design (minor changes to the canvas we built ourselves)

Table 5.3: AI Assistants used — Ch. 5

Chapter 6

Financials

6.1 Key Assumptions & 3-Year Projections

The projections are based on the following assumptions:

- **HW Sales**: gradual scale-up Y1→Y3 (GR → DE/NL → OEM volume) — figures in the table.
- **HW Pricing**: €89.99 (Y1–Y2), €79.99 (Y3) due to volume pricing.
- **Premium Subscribers**: conversion 40% (Y1) → ~63% (Y3) with the AI Assistant.
- **COGS HW**: €37.50/unit (Y1–Y2), €33.00 (Y3) with a 12% OEM volume discount.
- **COGS Cloud/LLM**: ~€3/subscriber/year (API + EU hosting), scaling up.
- **Fixed OPEX**: 2 founders' salaries, cloud, marketing, external contractors.

Metric	Year 1 (2026)	Year 2 (2027)	Year 3 (2028)
HW Units Sold	500	1,500	4,000
HW Revenue	€44,995	€134,985	€319,960
Premium Subscribers	200	800	2,500
Subscription Revenue	€24,000	€96,000	€330,000
Total Revenue	€68,995	€230,985	€649,960
COGS HW (€37.50 / €37.50 / €33.00)	€18,750	€56,250	€132,000
COGS Cloud/LLM (€3/sub/year)	€600	€2,400	€7,500
Total COGS	€19,350	€58,650	€139,500
Gross Profit	**€49,645**	**€172,335**	**€510,460**
Gross Margin	**~72%**	**~75%**	**~79%**
Fixed OPEX	€121,000	€154,500	€188,000
EBITDA	**−€71,355**	**€17,835**	**€322,460**

Table 6.1: Financial Projections 2026–2028 (Conservative Scenario)

Note: Y3 GM (~79%) reflects the classic IoT+SaaS logic: hardware margin improves with volume discounts, while the subscription (high-margin) gains a larger share of the revenue mix.

6.2 Unit Economics & Break-Even

The calculations below concern the Y1 blended customer (purchases the HW bundle + becomes a Premium subscriber). CAC is estimated at ~€55 based on paid social (Meta/TikTok: €2–5 CPC, ~2% CVR) and organic channels — realistic for consumer IoT in its initial phase [2].

Unit Economics	Break-Even Analysis
HW Bundle ASP: €89.99 (Y1)	Fixed OPEX Y1: €121,000
COGS per unit: €37.50 → HW CM ~58%	HW Contrib. Margin: ~€52/unit
Monthly Sub: €15/user (×12) = €180/year	Sub CM (~90%): ~€162/year
LTV (3Y Premium): €540/user	Blended CM (HW + 3Y Sub): ~€538/customer
Est. CAC: ~€55 (Y1, paid + organic)	Break-even: Y2 (based on P&L: EBITDA > 0)
LTV/CAC: ~10×	Break-even Month: ~18–20 **Y2** ✓

Table 6.2: Unit Economics & Break-Even Analysis

6.3 Download — Financial Spreadsheet

Google Sheets:
https://docs.google.com/spreadsheets/d/1tdKS6DRMxIYIVF4eUJHMc7P82owkoWOb/edit

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	—
Stavros Vlachos	8220019	**Gemini**: 3-year financial estimates based on the project's data

Table 6.3: AI Assistants used — Ch. 6

Chapter 7

The Team

7.1 Members & Roles

Designing DigiNova MoodLight requires combining two distinct technical disciplines that rarely coexist in an early-stage team: **embedded systems** (hardware / firmware) and **software development** (SW / UX). The team naturally covers both disciplines — with no external dependencies for the core product — and has proven in practice that it can deliver an end-to-end functional prototype (Arduino simulation + Android PWA) within a single academic term.

Member	Role	Responsibilities
Spilios Dimakopoulos Student ID: 8220035	Co-Founder / Hardware, Firmware & Prototype Lead	Arduino prototype (TinkerCAD), HR/HRV simulation, firmware (C++), stress-scoring algorithm, baseline calibration, system architecture diagram, building the HTML/CSS screens (SW prototype), developing & refining the JavaScript logic
Stavros Vlachos Student ID: 8220019	Co-Founder / Business, Design & Presentation Lead	UI/UX design (screen design, Figma wireframes), comparative hardware & competitive analysis, wiring diagram, market research, business model (BMC), financial projections, development of the business idea, creating & organizing the presentation (pitch deck)

Table 7.1: Team Members & Responsibilities

> **Why this team:** MoodLight is neither a purely hardware nor a purely software product — it is their integration. Spilios contributed the technical implementation (firmware, prototype, SW development), while Stavros led the business design, market analysis, and presentation — with shared contribution at both levels. This complementarity allowed the team to deliver an end-to-end solution with no external dependencies.

7.2 Gaps & External Partners

Gap	Solution / External Partner
iOS development	Freelance iOS developer or cross-platform (React Native)
PCB manufacturing	OEM (JLCPCB / PCBWay) for the custom MoodBox PCB
Clinical HRV validation	Medtech advisor / university partnership
CE marking / regulatory	External consultant for IoT medical device compliance
Digital marketing	Growth hacker / performance marketing agency
Professional UI/UX	Senior UX designer for a production-ready app

Table 7.2: Skill Gaps & Solutions

Chapter 8

Validation

8.1 Validation Strategy — From Hypothesis to Evidence

Before starting development, we set three critical hypotheses that needed to be confirmed:

1. **Demand Hypothesis:** Users aged 18–35 experience regular stress and are looking for a *passive* solution.
2. **Value Hypothesis:** The combination of automatic lighting and music is perceived as useful — not merely "nice to have."
3. **Privacy Hypothesis:** Recording biometric data would be a significant barrier to purchase.

To test them, we applied two successive validation phases: first **quantitative** (a broad survey) and then **qualitative** (hands-on user testing with the SW Prototype).

8.2 Phase 1 — Quantitative Market Research

8.2.1 Methodology & Sample Profile

In May 2026 we distributed an online questionnaire (6 questions) via Google Forms to **12 respondents** — students and young employees aged 18–35, i.e. exactly MoodLight's target customer. The small sample size doesn't allow for generalization, but it is sufficient for early-stage directional validation [1].

8.2.2 What We Found — and What Surprised Us

Hypothesis 1 — Fully confirmed. 83.4% of respondents reported experiencing stress regularly (50% "sometimes," 33.4% "often or almost always"). The finding confirms that the target problem is real and widespread.

Hypothesis 2 — Strongly confirmed. 75% already use music (50%) or lighting (25%) to relax — without anyone having asked them to. This means MoodLight isn't asking for new behavior: *it automates what they're already doing*. The concept rating came to **4.17/5** (75% rated it 4 or 5), with no one rating it below 3.

Hypothesis 3 — Overturned. This was the most important finding. We expected privacy concern to be the biggest barrier. Instead, concern about the privacy of biometric data scored only **2.50/5**, with 58.3% giving it a 1 or 2. This reshapes our strategy: privacy-first positioning is not a defensive move against a fear — it is a competitive advantage that builds trust with no critical resistance to overcome.

In the end, **91.7%** (11/12) expressed an intention to buy, with 41.6% accepting a price of €30–100.

8.2.3 What They Told Us Themselves

Participants responded positively: one described MoodLight as a concept that addressed a real need and asked for more music options; another said they would like to try it; and one simply called it an excellent idea. The request for more music options is already built into the roadmap as Spotify API integration (Premium tier).

8.3 Phase 2 — Qualitative User Testing

8.3.1 Methodology

In the second phase we presented the **SW Prototype** (see Ch. 4) to **5 users** matching the target profile: students and employees aged 21–28. We asked them to navigate the app freely with no instructions, perform three core actions (setup, reading the Stress Score, viewing history), and tell us spontaneously what they liked and what they would change.

User	Profile	Rating	Liked	To Improve
U1	CS student, 22 years old	5/5	Real-time score, nice concept	Music selection
U2	Employee, 28 years old	4/5	No need to remember to open it	Smartwatch compatibility
U3	Student, 21 years old	4/5	Quick setup, clear colors	Less visible ring
U4	Student, 24 years old	5/5	A friend also wanted to buy it	Weekly history
U5	Employee, 27 years old	3/5	Good idea for people with anxiety	€79 expensive with no trial

Table 8.1: Usability Testing Results

8.3.2 What We Learned from Each User

U2 summed up the core value proposition better than any description of ours could: "No need to remember to open it." This is exactly what sets MoodLight apart from every other wellness app.

U4 was the most revealing signal: they spontaneously mentioned that a friend also wanted to buy it after seeing the prototype. At the pre-launch stage, this is a sign of organic word-of-mouth — perhaps the strongest indicator of product-market fit.

U5, with the lowest rating (3/5), raised an actionable pricing problem: "€79 is expensive with no trial." This directly shapes the go-to-market plan: a **30-day trial with a money-back guarantee** will be considered at launch, to lower the perceived risk without lowering the price.

8.4 Validation Conclusions

Both phases consistently confirmed:

(a) **High demand for passive stress management:** 83.4% regularly exposed to stress, with no automatic tool available.

(b) **Acceptance of the freemium model:** 91.7% intention to buy; Premium is justified by AI insights and HRV trends.

(c) **Privacy-first positioning as a differentiator:** Low concern means that "Your data stays in Europe" builds trust without facing skepticism.

(d) **Need for a trial policy:** U5's feedback indicates that the perceived risk of buying with no trial needs to be addressed.

Users' spontaneous responses eloquently sum up the result: from "Excellent idea" and "I would like to try it" to the desire for "more music options" — a sign that the core value is understood and interest is extending to implementation details.

Next step: Full validation requires a clinical study with 50+ users after launch, a parallel CE marking submission, introducing a 30-day trial policy, and an Angel funding round of €150K.

Name	Student ID	AI Assistant & Use
Spilios Dimakopoulos	8220035	**Claude**: analysis and presentation of the validation results **Claude**: writing and formatting of this LaTeX document
Stavros Vlachos	8220019	**Claude**: analysis and presentation of the validation results

Table 8.2: AI Assistants used — Ch. 8

Bibliography

[1] World Health Organization. Guidelines on mental health at work. Technical report, World Health Organization, September 2022. ISBN 978-92-4-005794-7.

[2] IMARC Group. Wearable technology market report: Global industry trends, share, size, growth, opportunity and forecast 2025–2033. Technical report, IMARC Group, 2024.

[3] MarketsandMarkets Research. Wearable sensor market – global forecast to 2030. Technical report, MarketsandMarkets, 2025.

[4] Market Research Future. Stress management market research report – global forecast 2030. Technical report, Market Research Future, 2025.

[5] SNS Insider. Wearable health monitoring market size, share & trends analysis report, 2024–2032. Technical report, SNS Insider, 2024.

[6] Eurostat. Work and health statistics – mental health at work. Technical report, European Commission, Eurostat, 2024.

[7] Analog Devices (Maxim Integrated). *MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*. Analog Devices, Inc., 2018.

[8] Oura Health. Oura ring — smart ring health tracker, 2024. Accessed: June 2025.

[9] Signify (Philips Hue). Philips hue smart lighting system, 2024. Accessed: June 2025.

[10] Fred Shaffer and J. P. Ginsberg. An overview of heart rate variability metrics and norms. *Frontiers in Public Health*, 5:258, 2017.

[11] Hye-Geum Kim, Eun-Jin Cheon, Dai-Seg Bai, Young Hwan Lee, and Bon-Hoon Koo. Stress and heart rate variability: A meta-analysis and review of the literature. *Psychiatry Investigation*, 15(3):235–245, 2018.

[12] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data (GDPR), 2016.

Appendix 9

Full Firmware Source Code (MoodLight_Simulation.ino)

The code below is the complete firmware source code for the HW Prototype (see Ch. 3). It implements the Stress Score calculation algorithm, baseline calibration, and RGB LED / Piezo Buzzer control on the Arduino Uno R3.

```
// DigiNova MoodLight -- TinkerCAD Simulation
// Pot A0 = HR simulator | Pot A1 = HRV/RMSSD simulator
// RGB: D9=R, D10=G, D11=B | Buzzer: D8 | Status LED: D13

#define PIN_R      9
#define PIN_G      10
#define PIN_B      11
#define PIN_BUZZ   8
#define PIN_STATUS 13
#define HR_POT     A0
#define HRV_POT    A1
#define STRESS_THRESHOLD 65
#define BASELINE_SAMPLES 5

float baselineHR = 70.0, baselineRMSSD = 35.0;
int baselineCount = 0;
bool baselineReady = false;

float readHR() {
  // Pot A0: 0-1023 mapped to 40-180 BPM
  return map(analogRead(HR_POT), 0, 1023, 40, 180);
}

float readRMSSD() {
  // Pot A1: 0-1023 mapped to 5-80ms
  return map(analogRead(HRV_POT), 0, 1023, 5, 80);
}

int calcStress(float hr, float rmssd) {
  float refHR = baselineReady ? baselineHR : 70.0;
  float refR  = baselineReady ? baselineRMSSD : 35.0;
  float s = 50.0
    + ((hr - refHR) / max(refHR, 1.0f) * 30.0)
    + ((refR - rmssd) / max(refR, 1.0f) * 40.0);
  return constrain((int)s, 0, 100);
}

void setRGB(int r, int g, int b) {
  analogWrite(PIN_R, r);
  analogWrite(PIN_G, g);
  analogWrite(PIN_B, b);
}
```

```
43
44 void soundAlert(bool stress) {
45   if (stress) {
46     tone(PIN_BUZZ, 880, 100);
47     tone(PIN_BUZZ, 1100, 100);
48   } else {
49     noTone(PIN_BUZZ);
50   }
51 }
52
53 void setup() {
54   Serial.begin(9600);
55   pinMode(PIN_STATUS, OUTPUT);
56   for (int i = 0; i < 3; i++) {
57     setRGB(0, 0, 50); delay(200);
58     setRGB(0, 0, 0);  delay(200);
59   }
60   Serial.println("MoodLight TinkerCAD Ready");
61   Serial.println("Twist pots to simulate HR/HRV");
62 }
63
64 void loop() {
65   float hr    = readHR();
66   float rmssd = readRMSSD();
67
68   // Baseline calibration
69   if (!baselineReady) {
70     baselineHR = (baselineHR * baselineCount + hr)
71                  / (baselineCount + 1);
72     baselineRMSSD = (baselineRMSSD * baselineCount + rmssd)
73                     / (baselineCount + 1);
74     if (++baselineCount >= BASELINE_SAMPLES) {
75       baselineReady = true;
76       digitalWrite(PIN_STATUS, HIGH);
77       Serial.println("BASELINE_SET");
78     }
79   }
80
81   int score = calcStress(hr, rmssd);
82
83   // RGB color: green=calm -> yellow=moderate -> red=stress
84   if (score < 40) {
85     setRGB(0, score * 2, 0);
86   } else if (score < 65) {
87     int v = (score - 40) * 10;
88     setRGB(v, 200 - v/2, 0);
89   } else {
90     setRGB(200, 0, 0);
91   }
92
93   Serial.print("HR:"); Serial.print(hr, 0);
94   Serial.print(" RMSSD:"); Serial.print(rmssd, 0);
95   Serial.print(" SS:"); Serial.println(score);
96
97   if (score >= STRESS_THRESHOLD) {
98     Serial.println(">>> STRESS DETECTED -- Intervention ON");
99     soundAlert(true);
100  } else {
```

```
    soundAlert(false);
  }

  delay(800);
}
```

Listing 9.1: MoodLight_Simulation.ino — Complete firmware algorithm (HR/HRV stress scoring, RGB, buzzer, baseline calibration)

TinkerCAD Simulation Link:

https://www.tinkercad.com/things/0LYdJEJDMst-moodlight-mvp-simulation

YouTube Demo (Unlisted):

https://youtu.be/ffc-S6igFiQ

GitHub — Full Source Code:

https://github.com/SpiliosDimakopoulos/MoodLight

Appendix 10

Market Research Results

Online survey via Google Forms, May 2026, **12 responses**. The full interactive charts are available in the file `survey_dashboard.html` accompanying this report.

10.1 Key KPIs

Question / Metric	Result	Comment
Participants	12	Google Forms, May 2026
Concept usefulness (1–5)	**4.17/5**	75% rated it 4 or 5
Privacy concern (1–5)	**2.50/5**	58% rated it 1–2
Intention to buy	**91.7%**	11 out of 12 respondents

Table 10.1: Aggregate Survey Results

10.2 Analysis by Question

Answer	N	%
Rarely	2	16.7%
Sometimes	6	50.0%
Often	2	16.7%
Almost always	2	16.7%
Total	**12**	**100%**

Table 10.2: Q1 — Stress Frequency

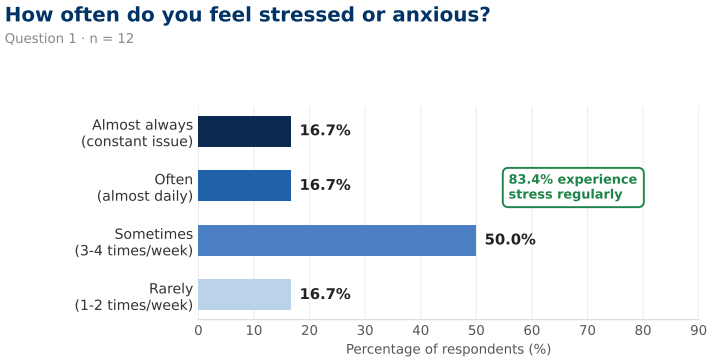

Figure 10.1: Q1 — Stress frequency (n=12): **83.4%** regularly

Answer	N	%
Music / sounds	6	50.0%
Lighting / candles	3	25.0%
Social/Netflix	2	16.7%
Exercise	1	8.3%
Total	**12**	**100%**

Table 10.3: Q2 — Relaxation Method

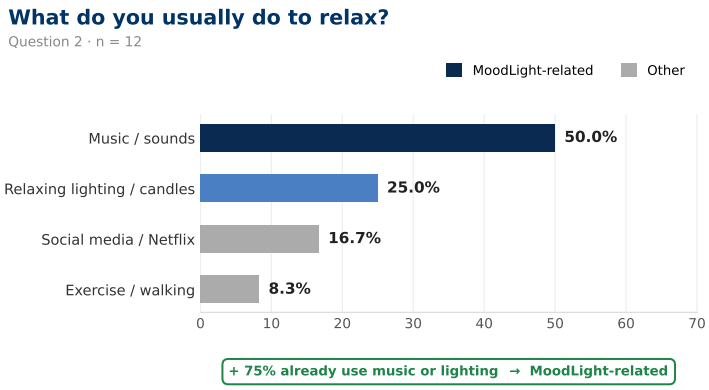

Figure 10.2: Q2 — **75%** already use music or lighting

Score	N	%	Meaning
1	0	0%	Not at all
2	0	0%	
3	3	25%	Neutral
4	4	33.3%	Quite
5	5	41.7%	Very
Avg.			**4.17/5**

Table 10.4: Q3 — Usefulness (1–5)

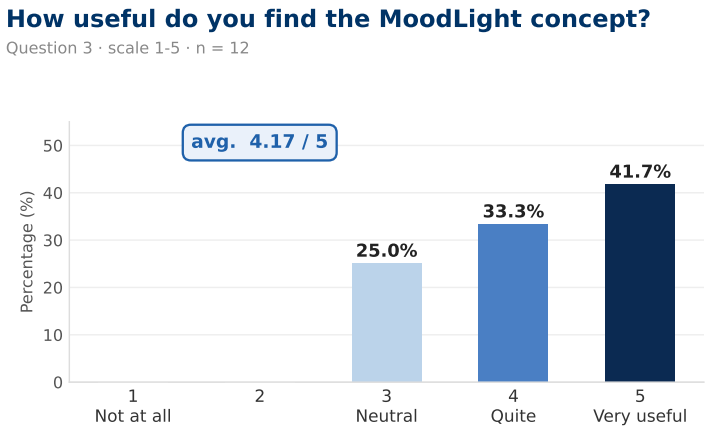

Figure 10.3: Q3 — avg. **4.17/5**, no one <3

Score	N	%	Meaning
1	4	33.3%	Not at all
2	3	25.0%	Low
3	2	16.7%	Moderate
4	1	8.3%	High
5	2	16.7%	Very high
Avg.			**2.50/5**

Table 10.5: Q4 — Privacy Concern

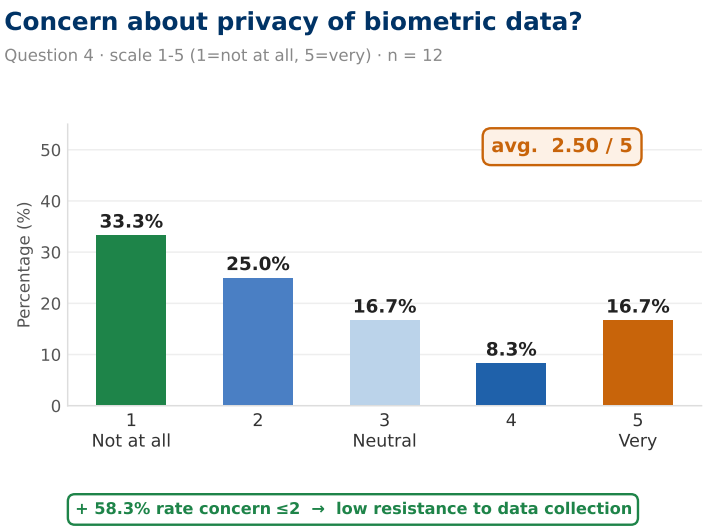

Figure 10.4: Q4 — avg. **2.50/5**, **58.3%** rated 1–2

Price	N	%
Up to €30	6	50.0%
€30–€70	4	33.3%
€70–€100	1	8.3%
Would not buy	1	8.3%
Total	**12**	**100%**

Table 10.6: Q5 — Willingness to Pay

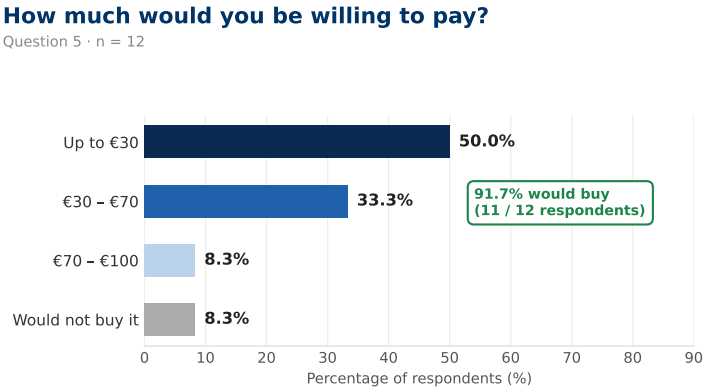

Figure 10.5: Q5 — **91.7%** intention to buy

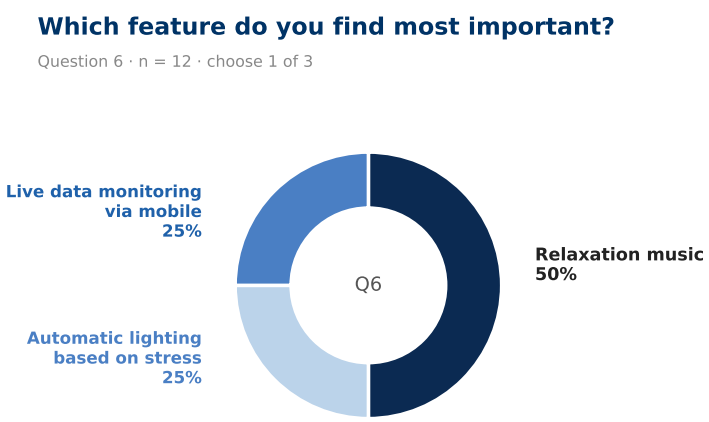

Figure 10.6: Q6 — Music ranks first (**50%**)

Feature	N	%
Relaxation music	6	50.0%
Automatic lighting	3	25.0%
Live tracking	3	25.0%
Total	**12**	**100%**

Table 10.7: Q6 — Most Important Feature

10.3 Qualitative Comments

The qualitative comments are presented in Ch. 8 (§ Validation Conclusions).