JEGYZŐKÖNYV

Web technológiák 1

Méhészet weboldala

Készítette: Baba Levente

Neptunkód: HLFA5R

Dátum: 2025. december 02.

Miskolc, 2025

Tartalomjegyzék

Bevezetés

A Webtechnológiák 1 tárgy beadandó feladataként egy teljes, többboldalas webalkalmazást készítettem „Zümipapa Méhészet” témában. A cél egy olyan, modern felépítésű, reszponzív honlap volt, amely egyrészt bemutatja egy családi kézműves méhészet szolgáltatásait és termékeit, másrészt demonstrálja a felhasználókezelés alapjait egy külön Node.js alapú backend segítségével. A frontend részt a követelményeknek megfelelően tisztán HTML5, CSS és natív JavaScript felhasználásával valósítottam meg, külön külső frameworkök nélkül. A főoldal (index.html) áttekintést ad a méhészetről, kiemeli a fő üzenetet („A mézfogyasztás sohase legyen terhes kötelesség...”), tartalmaz egy „hero” blokkot, hírek és akciók szekciót, illetve gyors átjárást biztosít az aloldalakhoz, mint például az ismertető, a terméklista és a mézes receptek.

A honlap több tematikus aloldalra tagolódik. A „A mézről” oldal hosszabb, ismeretterjesztő szöveggel mutatja be a méz történetét, kristályosodását, egészségügyi hatásait, valamint a hamis mézek felismerésének szempontjait, ezzel tartalmi szempontból is értékes, jól strukturált anyagot szolgáltat.

A „Méhes rímek” oldalon gyerekeknek szóló versek és mondókák jelennek meg, illusztrációkkal kiegészítve, amelyek a méhekkel és a szorgos munkával kapcsolatosak, így a honlap családbarát, játékos tartalommal is rendelkezik.

A „Képek” aloldal egy méhészetről szóló, saját videó beágyazását tartalmazza: egy HTML5 video elemet használtam, amelyhez teljesen egyedi vezérlőfelületet készítettem (lejátszás/szünet, 10 másodperces előre- és visszatekerés, némítás, hangerő-csúszka, idő kijelzés). Emellett további aloldalak (pl. terméklista, receptek, kapcsolat, adatvédelem) biztosítják, hogy a honlap felépítése egy valós, teljes értékű kisvállalkozási weboldal struktúráját tükrözze.

A frontend egységes megjelenéséről a központi CSS állomány gondoskodik: ebben definiáltam a színpalettát (méz, levélzöld, világos háttér), a rácselrendezéseket, a kártya-szerű dobozokat, a ragadós fejléct, a táblázatok és űrlapok stílusát, valamint a videólejátszóhoz tartozó elemek kinézetét.

Külön figyelmet fordítottam arra, hogy a navigáció kis kijelzőn hamburger menüre váltson, és hogy a tipográfia, margók, gombok mérete mobilon is jól használható maradjon. A JavaScript kód egyrészt a felhasználói élményt javítja (jelszó mutatása/elrejtése, aktuális évszám automatikus kiírása a láblécben, téma-beállítás „Windows szerinti” és fix világos mód között), másrészt az alkalmazás üzleti logikáját is megvalósítja, például a bejelentkezés, regisztráció és a felhasználólista dinamikus betöltése során.

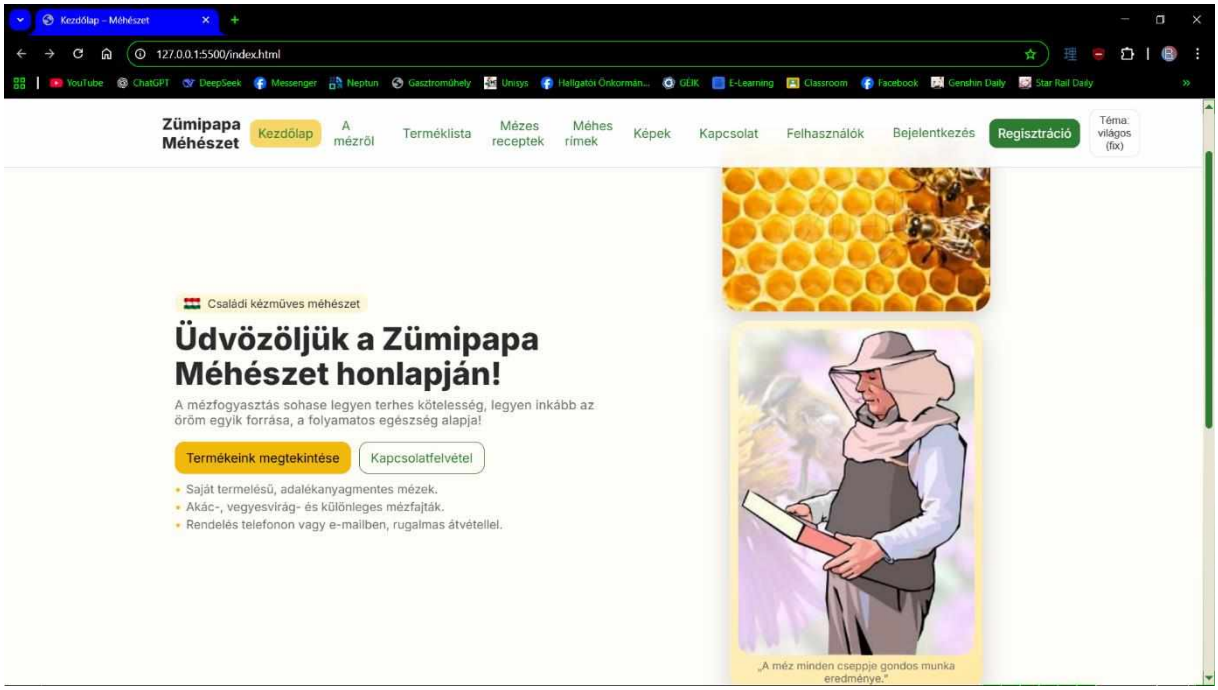
A projekt másik fontos része a külön Node.js alapú backend, amely REST-szerű HTTPS végpontokon keresztül szolgálja ki a frontend által használt adatokat. A szerver Express keretrendszert használ, önaláírt tanúsítvánnyal indul HTTPS módban, és JSON formátumú, fájlalapú „adatbázisban” tárolja a regisztrált felhasználók adatait.
A /api/auth/register végpont feladata az új felhasználók regisztrációja: validálja a beküldött adatokat, ellenőrzi, hogy az e-mail cím egyedi-e, majd a jelszót sózott-hasheléssel elmenti, így nem tárolódik sehol tiszta szöveges jelszó. A /api/auth/login végpont a bejelentkezést valósítja meg, a korábban eltárolt hash alapján ellenőrzi a jelszót, és csak a nem érzékeny felhasználói adatokat adja vissza. A /api/users és /api/users/:id végpontok a felhasználók listázását, egyedi lekérdezését, illetve törlését teszik lehetővé. A frontend a bejelentkezett felhasználó adatait localStorage-ben tárolja, ezek alapján frissíti a fejléc „Bejelentkezve: …” állapotát, és a „Felhasználók” oldalon dinamikusan tölti be és jeleníti meg a felhasználókat, beleértve a törlés lehetőségét is.
Összességében a feladat célja egy olyan webes alkalmazás létrehozása volt, amely egyszerre mutatja be a klasszikus, tartalomközpontú honlapkészítést és a modern webes technológiák (HTTPS, JSON alapú REST API, kliensoldali fetch hívások, alapvető felhasználókezelés és jelszóhash-elés) használatát. A választott méhészet-téma jó alapot adott arra, hogy a különböző funkciókat (tartalomoldalak, videólejátszó, regisztráció/bejelentkezés, felhasználólista) egy koherens, valószerű, gyakorlati példában egyesítsem.

Főoldal

A főoldal egy letisztult, reszponzív kezdőlap, amely elsőként egy fix fejlécet jelenít meg márkával és főmenüvel, ahonnan minden fontos aloldal (ismertető, terméklista, receptek, versek, képek, kapcsolat, felhasználók) elérhető. A központi „hero” szekcióban egy magyar zászlós felirat („Családi kézműves méhészet”), egy nagy címsor és egy kiemelt mottó foglalja össze a méhészet alapüzenetét, mellette pedig két elsődleges gomb irányítja a látogatót a termékekhez és a kapcsolatfelvételhez. A hero rész jobb oldalán nagy mézes és méhészes fotók jelennek meg, amelyek vizuálisan is erősítik a kézműves jellegű, bizalmi hangulatot. A főblokkok alatt három kártya vezeti tovább a látogatót: egy ismertető oldalra („Amit a mézről tudni kell”), a teljes terméklistára, illetve a mézes recepteket tartalmazó oldalra. Végül egy „Hírek és akciók” szekció emeli ki az aktuális eseményeket (pl. friss akácméz érkezése, kóstoló időpontja), így a főoldal egyszerre szolgál bemutatkozó, navigációs és információs központként. A láblécben a szerzői jogi információ mellett a kapcsolat és az adatvédelmi oldal érhető el, a JavaScript pedig automatikusan az aktuális évszámot jeleníti meg.

Példakód a főoldal hero szekciójából:

```
<section class="hero hero-layout">
  <div class="hero-text">
    <div class="hero-kicker">
      <img src="assets/images/flag-hu.gif" alt="Magyar zászló" class="hero-flag">
      <span>Családi kézműves méhészet</span>
    </div>

    <h1>Üdvözöljük a Zümipapa Méhészet honlapján!</h1>

    <p class="tagline">
      A mézfogyasztás sohase legyen terhes kötelesség, legyen inkább az öröm
egyik forrása,
      a folyamatos egészség alapja!
    </p>

    <div class="hero-actions">
      <a class="btn btn-secondary" href="/pages/products.html">Termékeink
megtekintése</a>
      <a class="btn btn-outline" href="/pages/contact.html">Kapcsolatfelvétel</a>
    </div>

    <ul class="hero-list">
      <li>Saját termelésű, adalékanyagmentes mézek.</li>
      <li>Akác-, vegyesvirág- és különleges mézfajták.</li>
      <li>Rendelés telefonon vagy e-mailben, rugalmas átvétellel.</li>
    </ul>
  </div>

  <div class="hero-media">
    <div class="hero-media-bg">
      <img src="assets/images/header-honey.jpg" alt="" loading="lazy">
    </div>
    <figure class="hero-figure">
      <img src="assets/images/beekeeper.jpg" alt="Méhész a méhek között"
loading="lazy">
      <figcaption>„A méz minden cseppje gondos munka
eredménye.”</figcaption>
    </figure>
  </div>
</section>
```



1. ábra - Főoldal

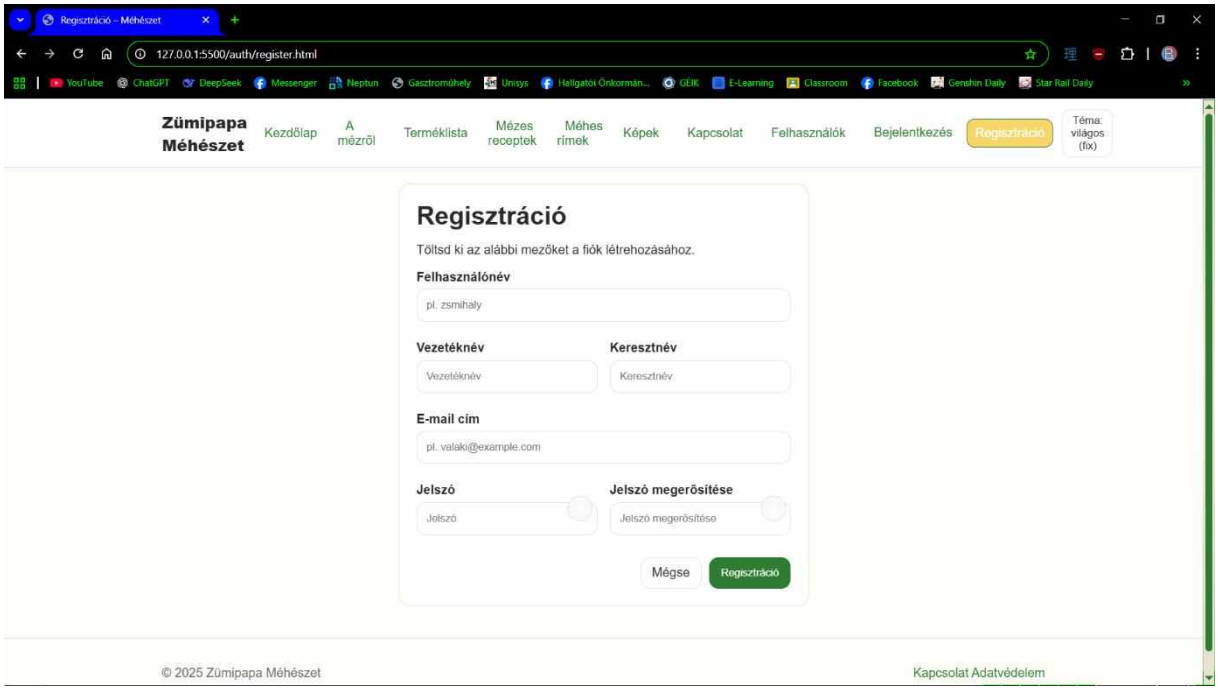
Felhasználókezelés

Az alkalmazás tartalmaz egy egyszerű, de teljes értékű felhasználókezelést, amely regisztrációból, bejelentkezésből és a regisztrált felhasználók listázásából, törléséből áll. A frontend oldalon külön „Regisztráció" és „Bejelentkezés" űrlap található, ahol a felhasználó megadja a szükséges adatait (felhasználónév, név, e-mail, jelszó), a JavaScript pedig kliensoldalon ellenőrzi a mezők kitöltését és azt, hogy a két jelszómező egyezik. A jelszót a böngészőben először SHA-256 algoritmussal hash-elem, majd Base64 formátumban küldöm el a Node.js alapú HTTPS backend felé, ahol a szerver további validációt végez, ellenőrzi, hogy az e-mail cím egyedi-e, és egy külön sózott-hashelést alkalmaz, így a jelszó sehol nem tárolódik tiszta szövegként. Sikeres regisztráció vagy bejelentkezés után a backend egy „megtisztított" felhasználói objektumot ad vissza (jelszó nélkül), amelyet a frontend localStorage-ben tárol, és ennek alapján frissíti a fejlécben a „Bejelentkezve: …" állapotot. A „Felhasználók" aloldalon a kliens a /api/users végpontot hívva betölti a regisztrált felhasználók listáját, táblázatban jeleníti meg őket, és egy gombbal lehetőség van egyes felhasználók törlésére; ha a bejelentkezett felhasználó saját magát törli, a rendszer automatikusan kijelentkezteti.

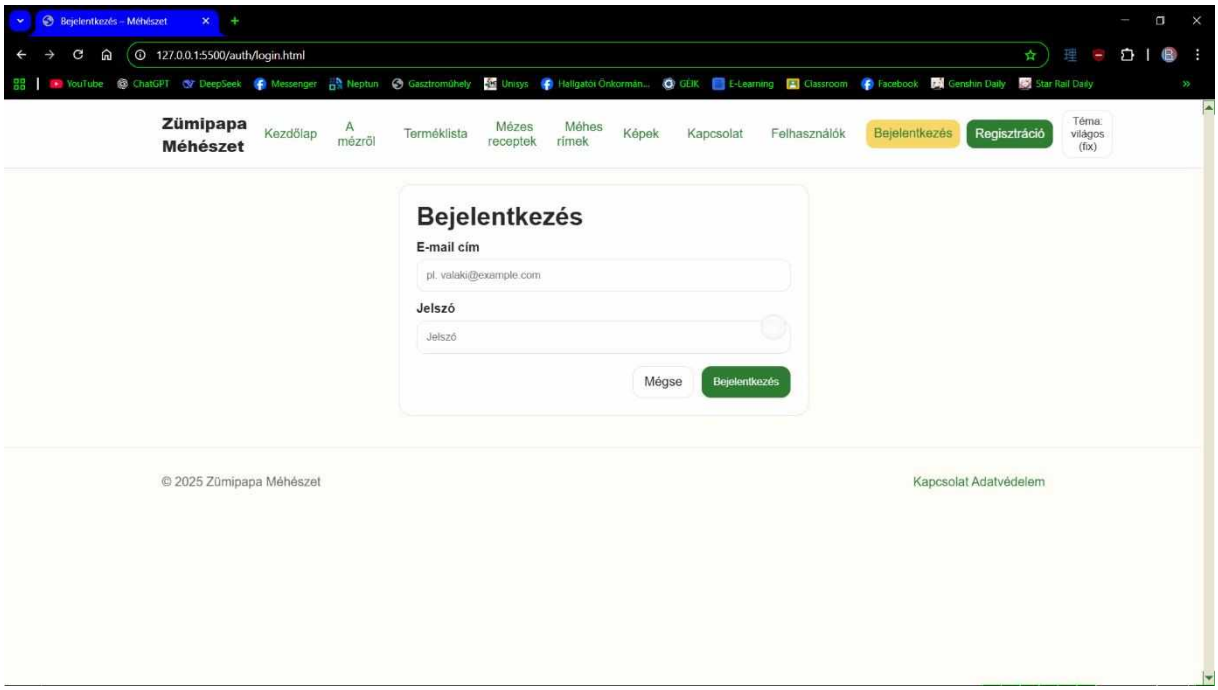
Példakód a backend regisztrációs végpontjából:

```
app.post('/api/auth/register', (req, res) => {
  const { valid, errors } = validateRegisterInput(req.body);

  if (!valid) {
    return res.status(400).json({ errors });
  }

  const username = req.body.username.trim();
  const firstName = req.body.firstName.trim();
  const lastName = req.body.lastName.trim();
```

```
    const email = req.body.email.trim();
    const password = req.body.password;

    const existing = findUserByEmail(email);
    if (existing) {
        return res.status(409).json({ message: 'User with this email already exists' });
    }

    let passwordData;
    try {
        passwordData = createPasswordHash(password);
    } catch {
        return res.status(400).json({ message: 'Invalid password format' });
    }

    const user = {
        id: generateId(),
        email,
        username,
        firstName,
        lastName,
        passwordHash: passwordData.hash,
        passwordSalt: passwordData.salt
    };

    addUser(user);

    const { passwordHash, passwordSalt, ...publicUser } = user;
    return res.status(201).json(publicUser);
});
```



2. ábra – Regisztráció



3. ábra - Bejelentkezés

Videókezelés

Az alkalmazás a „Képek” aloldalon egy HTML5 alapú videólejátszót tartalmaz, amely egy méhekről szóló, főként kisebbeknek szánt ismeretterjesztő videót jelenít meg. A lejátszóhoz nem a böngésző alapértelmezett vezérlősávját használom, hanem teljesen egyedi vezérlőket: külön gombokkal indítható és megállítható a lejátszás, 10 másodperces lépésekben előre- és vissza lehet tekerni, némítási funkció és folyamatosan állítható hangerő-csúszka is rendelkezésre áll. A JavaScript kód a data-* attribútumokkal megjelölt elemeket választja ki, és eseménykezelőkkel köti össze a gombokat a videó objektum megfelelő metódusaival (play,

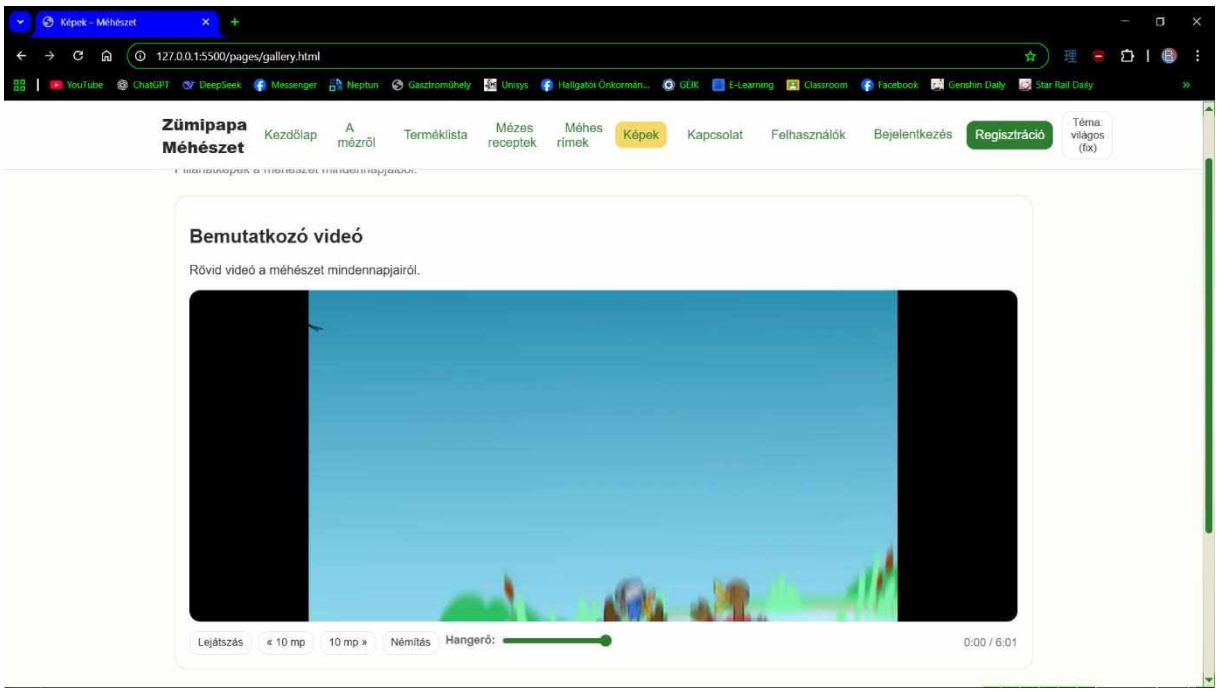
pause, currentTime, volume, muted). A lejátszó alatt egy időzítő felirat jelenik meg, amely a currentTime és a duration alapján perc:másodperc formában mutatja az aktuális pozíciót és a videó teljes hosszát, a metaadatok beolvasása után automatikusan frissítve. A megoldás szemlélteti a HTML5 video elem, az eseménykezelés és a DOM-manipuláció kombinálását egy valós funkció, a testreszabott videókezelés megvalósítására.

Példakód a videókezelés JavaScript részéből (main.js):

```
function initGalleryVideo() {
  const wrapper = document.querySelector('[data-video-player="gallery"]');
  if (!wrapper) return;

  const video = wrapper.querySelector('video');
  if (!video) return;

  const playPauseBtn = wrapper.querySelector('[data-video-control="play-pause"]');
  const backwardBtn = wrapper.querySelector('[data-video-control="backward"]');
  const forwardBtn = wrapper.querySelector('[data-video-control="forward"]');
  const muteBtn = wrapper.querySelector('[data-video-control="mute"]');
  const volumeSlider = wrapper.querySelector('[data-video-control="volume"]');
  const timeLabel = wrapper.querySelector('[data-video-time]');

  function formatTime(seconds) {
    if (!isFinite(seconds)) return '0:00';
    seconds = Math.max(0, Math.floor(seconds));
    const m = Math.floor(seconds / 60);
    const s = seconds % 60;
    return m + ':' + (s < 10 ? '0' + s : s);
  }

  function updateTime() {
    if (!timeLabel) return;
    const current = video.currentTime || 0;
    const total = isFinite(video.duration) ? video.duration : 0;
    timeLabel.textContent = formatTime(current) + ' / ' + formatTime(total);
  }

  if (playPauseBtn) {
    playPauseBtn.addEventListener('click', () => {
      if (video.paused || video.ended) {
        video.play();
      } else {
        video.pause();
      }
```

```
  });

  video.addEventListener('play', () => {
    playPauseBtn.textContent = 'Szünet';
  });

  video.addEventListener('pause', () => {
    playPauseBtn.textContent = 'Lejátszás';
  });
}

if (backwardBtn) {
  backwardBtn.addEventListener('click', () => {
    const target = (video.currentTime || 0) - 10;
    video.currentTime = Math.max(0, target);
  });
}

if (forwardBtn) {
  forwardBtn.addEventListener('click', () => {
    const duration = isFinite(video.duration) ? video.duration : 0;
    const target = (video.currentTime || 0) + 10;
    video.currentTime = duration ? Math.min(duration, target) : target;
  });
}

if (muteBtn) {
  muteBtn.addEventListener('click', () => {
    video.muted = !video.muted;
    muteBtn.textContent = video.muted ? 'Hang bekapcs.' : 'Némítás';
  });
}

if (volumeSlider) {
  volumeSlider.addEventListener('input', () => {
    const value = parseFloat(volumeSlider.value);
    if (!isNaN(value)) {
      video.volume = value;
      if (value === 0) {
        video.muted = true;
        if (muteBtn) {
```

```
                muteBtn.textContent = 'Hang bekapcs.';
            }
        } else {
            video.muted = false;
            if (muteBtn) {
                muteBtn.textContent = 'Némítás';
            }
        }
    }
  });
}

video.addEventListener('timeupdate', updateTime);
video.addEventListener('loadedmetadata', updateTime);
updateTime();
}
```



4. ábra – Videó megjelenítése és videó lejátszás irányítása