1) Közös helper függvények külön fájlban (lib.php)

Cél: függvények kiszervezése külön fájlba, újrafelhasználás.

Feladat:

1. Hozzon létre egy lib.php fájlt, amely legalább 3 függvényt tartalmaz:
    - HTML-escape (biztonságos kiírás),
    - pénzformázás forintban,
    - egy olyan segédfüggvény, ami egy egész számot egy tartományba kényszerít (pl. 0-999).
2. Készítsen egy index.php oldalt, amely a lib.php-t betölti, majd egy GET-es űrlapból beolvas:
    - egységár,
    - mennyiség.
3. Számolja ki az összegárat, és jelenítse meg táblázatban:
    - egységár (formázva),
    - mennyiség (tartományra korlátozva),
    - összesen (formázva).
4. Gondoskodjon arról, hogy hibás input esetén se omoljon össze a program (pl. nem szám esetén 0).

---

2) Kalkulátor: műveletek külön modulban (calc.php)

Cél: üzleti logika (műveletek) elkülönítése a felülettől.

Feladat:

1. Hozzon létre calc.php fájlt, amely függvényeket tartalmaz:
    - összeadás,
    - kivonás,
    - szorzás,
    - osztás (0-val osztás esetén kezelje hibával).
2. A index.php oldalon készítsen űrlapot:
    - A szám,
    - művelet (select),

- B szám.

3. Validálja az inputokat szerveroldalon, és hibánál mezőnként jelezzen.
4. Siker esetén számoljon a calc.php függvényeinek meghívásával, és írja ki az eredményt.

---

3) Validációs könyvtár (validators.php)

Cél: ismétlődő validációk kiszervezése újrafelhasználható függvényekbe.

Feladat:

1. Hozzon létre validators.php fájlt, amely legalább ezeket tudja:
   - kötelező mező ellenőrzése,
   - e-mail validáció,
   - egész szám tartomány ellenőrzése (min-max).
2. Készítsen űrlapot (név, e-mail, életkor).
3. Az index.php csak a validátor függvényeket hívja, ne legyen benne duplikált validációs logika.
4. Siker esetén jelenítsen meg összefoglalót a beküldött adatokról.

---

4) Header/Footer sablonok külön fájlban (templates)

Cél: közös HTML elemek kiszervezése, include használata.

Feladat:

1. Hozzon létre egy templates/ mappát.
2. Készítsen benne:
   - header.php (HTML head + oldal cím),
   - footer.php (záró elemek).
3. Az index.php ezeket az elemeket include-olja.
4. Legyen egy egyszerű űrlap (pl. téma választás selectből), és jelenítse meg az eredményt a lapon.

---

5) Mini router: több oldal egy projektben (pages mappa)

Cél: több „aloldal” kezelése egy projekten belül.

Feladat:

1. Hozzon létre egy pages/ mappát három oldallal:
    - home,
    - about,
    - contact.
2. Az index.php egy GET paraméter (pl. page) alapján döntse el, melyik oldalt tölti be.
3. Csak az engedélyezett oldalak legyenek betölthetők (whitelist).
4. Legyen navigációs menü a három oldal között.

---

6) Form feldolgozás: controller és view szétválasztása

Cél: „feldolgozás” és „megjelenítés” szétválasztása több fájlba.

Feladat:

1. Készítsen controller.php fájlt, amely:
    - beolvassa és validálja a POST adatokat,
    - visszaadja az adatokat és hibákat (pl. tömbben).
2. Készítsen view.php fájlt, amely:
    - megjeleníti az űrlapot,
    - kiírja a hibákat és visszatölti az értékeket.
3. Az index.php csak összefogja a folyamatot:
    - controller meghívása,
    - view betöltése,
    - siker esetén összegzés megjelenítése.
4. Téma: „Kapcsolat” űrlap (név, e-mail, üzenet).

---

7) Újrafelhasználható JSON tároló modul (storage.php)

Cél: egy egyszerű fájl-alapú perzisztencia réteg létrehozása.

Feladat:

1. Készítsen storage.php fájlt, amely tud:
    - JSON betöltést (fájlból tömbbe),
    - JSON mentést (tömbből fájlba),
    - új elem hozzáfűzését (append).
2. Az index.php oldalon készítsen egyszerű „jegyzet mentő” funkciót:
    - 1 mező: szöveg (max 200 karakter),
    - mentés után PRG,
    - listázás táblázatban.
3. Legyen „törlés” funkció (fájl törlése).

---

8) Logger modul (logger.php) több szinttel

Cél: közös naplózási logika külön modulban.

Feladat:

1. Készítsen logger.php fájlt, amely egy függvényen keresztül naplóz:
    - időbélyeg,
    - log szint (INFO/WARN/ERROR),
    - üzenet,
    - opcionális context (pl. IP).
2. Az oldalon legyen űrlap:
    - szint kiválasztása,
    - üzenet.
3. Mentés után PRG, majd listázza az utolsó 30 sort.
4. Legyen log törlése funkció.

---

9) Mini MVC felosztás (model / controller / views)

Cél: alap rétegzett gondolkodás: adat, logika, megjelenítés.

Feladat:

1. Hozzon létre model.php fájlt, ami ad egy terméklistát (tömb).
2. Hozzon létre controller.php fájlt, ami:
    - lekéri az adatot a modellből,
    - kiszámolja az összesítést (pl. összeg),
    - előállítja a megjelenítendő tartalmat.
3. Hozzon létre views/layout.php fájlt, ami egységes HTML keretbe teszi a tartalmat.
4. Az index.php csak a „belépési pont” legyen.

---

10) Moduláris számolás + táblázat (összetett, több fájl)

Cél: nagyobb feladatban több függvény és több fájl együttműködtetése.

Feladat:

1. Kérjen be egy N értéket (1-50).
2. Készítsen külön stats.php fájlt, amely függvényekre bontva:
    - generálja az 1..N sorozatot,
    - kiszámol egy sorhoz értékeket (i, i^2, i^3, $\sqrt{i}$),
    - összegzést készít (pl. i^2 és i^3 átlag).
3. Az index.php:
    - validálja N-t,
    - meghívja a stats.php függvényeit,
    - táblázatban megjeleníti az eredményeket,
    - és a táblázat végén kiírja az átlagokat.
4. A HTML keretet templates/header.php + templates/footer.php fájlokból illessze be.