Web Programming

Son Hafta: Entegrasyon

HTML • CSS • JS • PHP • MySQL

Son Hafta: 10 Entegre Mini Site

HTML • CSS • JavaScript • PHP • MySQL bir arada

Dersin amacı

Öğrenciler artık tek tek teknolojileri değil, küçük ama gerçekçi bir web sitesinin parçalarının nasıl birlikte çalıştığını görecek. Her örnek bir kullanıcı hikayesi, veritabanı, form, PHP işlem dosyası ve çıktı ekranı içerir.

Bu hafta öğrencinin görmesi gereken bağ

1) HTML form veriyi toplar
2) JS küçük kontrol/etkileşim sağlar
3) PHP isteği karşılar
4) SQL veriyi okur/yazar
5) HTML çıktısı yeniden oluşur

10 gerçek dünya mini-site örneği

1. Etkinlik Kayıt Sistemi
2. Randevu Takip Sistemi
3. Destek Talep Paneli
4. Ürün Stok ve Sepet Önizleme
5. Kütüphane Rezervasyon Sistemi
6. Restoran Menü ve Sipariş Taslağı
7. Gider Takip Uygulaması
8. Ders Geri Bildirim Formu
9. Mini Blog ve Yorum Paneli
10. Kayıp-Eşya İlan Panosu

Ortak mimari: aynı akış, farklı mini siteler

Tek sayfalık “demo” değil; her mini site küçük bir tam yığın web uygulaması gibi düşünülür.

HTML

Formlar, tablolar, linkler ve sayfa iskeleti

CSS

Sadece düzen, okunabilirlik, uyarı/başarı görünümü

JavaScript

Anlık kontrol, toplam hesaplama, confirm/alert

PHP

GET/POST okuma, validation, echo ile çıktı

SQL

SELECT, INSERT, UPDATE, DELETE, JOIN, GROUP BY

Sunum mantığı

Her mini site 9 slaytta işlenir: hikaye → DB → dosya akışı → HTML/PHP → CSS → JS → SELECT → POST/UPDATE/DELETE → entegrasyon testi.

Ortak klasör yapısı ve dosya sorumlulukları

KLASÖR YAPISI

```
mini-site/
  db.php                # PDO bağlantısı
  index.php             # listeleme / ana ekran
  save.php              # POST işlemi
  update.php            # UPDATE veya status
değişimi
  delete.php            # gerekli örneklerde
silme
  assets/
    style.css           # sade görünüm
    app.js              # küçük etkileşim veya
form
      kontrolü
```

index.php

DB'den veri çeker, HTML çıktısını üretir ve form/linkleri gösterir.

save/update/delete.php

Formdan gelen veriyi doğrular, prepared statement ile SQL çalıştırır, sonra kullanıcıyı geri yönlendirir.

assets/app.js

Sayfa yenilenmeden küçük kontroller yapar; asıl güvenlik ve kayıt işi yine PHP tarafındadır.

Ortak PHP bağlantısı: db.php

db.php

```php
<?php
$host = "localhost";
$dbname = "mini_sites";
$user = "root";
$pass = "";

$dsn =
"mysql:host=$host;dbname=$dbname;charset=utf8mb4";
$pdo = new PDO($dsn, $user, $pass);
$pdo->setAttribute(PDO::ATTR_ERRMODE,
  PDO::ERRMODE_EXCEPTION);
$pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
  PDO::FETCH_ASSOC);
?>
```

Neden PDO?

Aynı ders içinde güvenli sorgu mantığını göstermek için prepared statement kullanılır.

GÜVENLİ SELECT

```php
$stmt = $pdo->prepare(
  "SELECT * FROM products WHERE name LIKE ?"
);
$stmt->execute(["%" . $_GET["q"] . "%"]);

foreach ($stmt as $row) {
  echo htmlspecialchars($row["name"]);
}
```

Ortak request/response döngüsü

HTML İÇİNDE PHP

```
<!-- index.php -->
<form method="post" action="save.php">
  <input name="title" required>
  <button>Save</button>
</form>

<?php
foreach ($rows as $row) {
  echo htmlspecialchars($row["title"]);
}
?>
```

1. Kullanıcı formu doldurur

HTML input değerleri tarayıcıdan sunucuya gider.

2. PHP veriyi okur

$_GET veya $_POST ile gelen veri trim edilir ve kontrol edilir.

3. SQL çalışır

SELECT/INSERT/UPDATE/DELETE prepared statement ile çalışır.

4. Çıktı tekrar HTML olur

PHP echo ile güvenli metni HTML içine basar.

Mini Site 1: Etkinlik Kayıt Sistemi

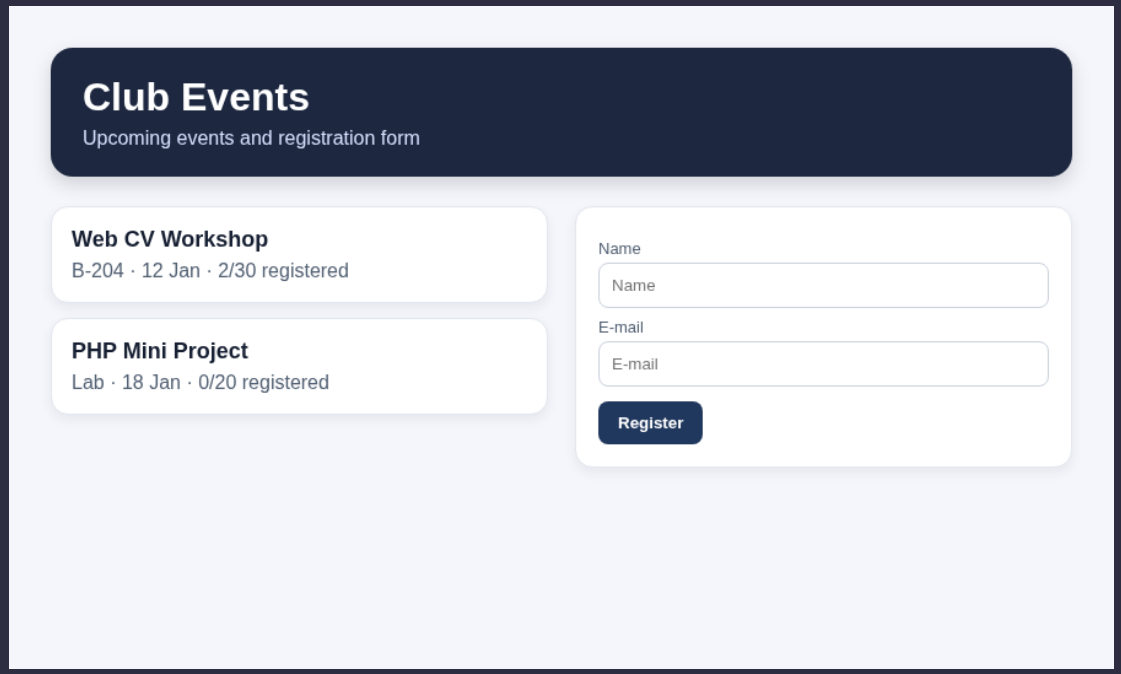
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Bir öğrenci kulübü yaklaşan etkinlikleri listeler. Öğrenciler kontenjanı dolmadan kayıt formunu doldurur; yönetici hangi etkinliğe kaç kişinin kaydolduğunu görür.

Öğrenci bu mini sitede neyi birleştirir?

- GET ile kategori/tarih filtresi
- POST ile kayıt alma
- JOIN + COUNT ile kontenjan takibi
- Form doğrulama ve güvenli çıktı



Beklenen çıktı: ana ekran / kullanıcı formu

Etkinlik Kayıt Sistemi: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE events (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(100), event_date DATE,
  room VARCHAR(50), capacity INT
);
CREATE TABLE registrations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  event_id INT, student_name VARCHAR(80), email
    VARCHAR(120),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (event_id) REFERENCES events(id)
);
```

DB ÖRNEK VERİ

```
events
id | title             | event_date | room | capacity
1  | Web CV Workshop   | 2026-01-12 | B-204| 30
2  | PHP Mini Project  | 2026-01-18 | Lab  | 20

registrations
id | event_id | student_name | email
1  | 1        | Ada Yılmaz   | ada@mail.com
2  | 1        | Mert Kaya    | mert@mail.com
```

Etkinlik Kayıt Sistemi: Dosya akışı

Dosyalar ve görevleri

- index.php: etkinlik listesi ve kayıt formu
- register.php: POST verisini doğrular ve kaydeder
- admin.php: kayıt sayıları ve katılımcılar
- db.php + assets/app.js + assets/style.css

REQUEST AKIŞI

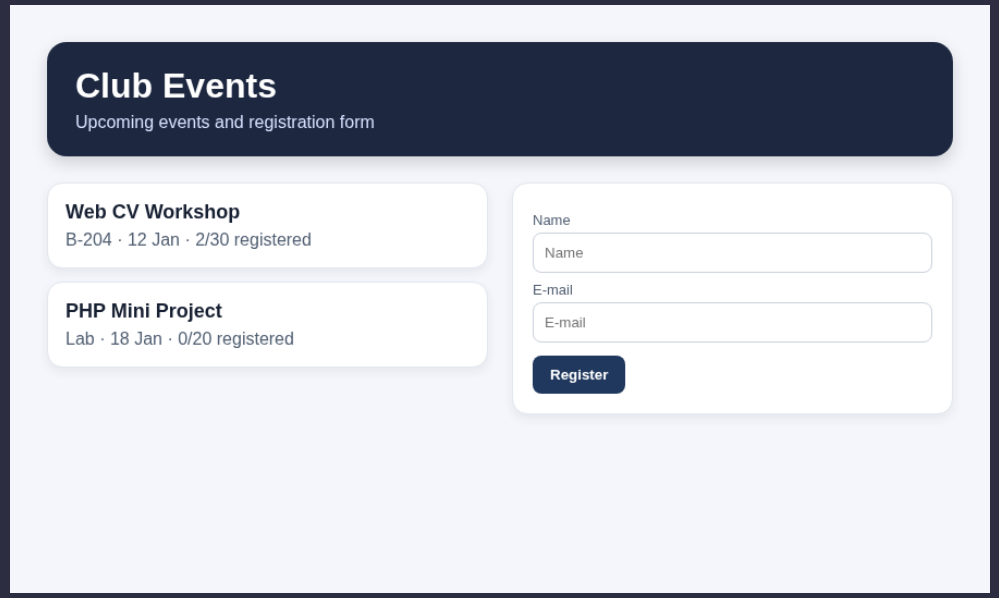
```
GET /index.php?q=php -> filtrelenmiş etkinlikler
POST /register.php -> registrations tablosuna
INSERT
GET /admin.php -> events + registrations JOIN
raporu
```

Etkinlik Kayıt Sistemi: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<?php include "db.php"; ?>
<h1>Club Events</h1>
<form method="get" action="index.php">
  <input name="q" placeholder="Search event">
  <button>Search</button>
</form>
<?php foreach ($events as $event): ?>
  <article>
    <?php echo htmlspecialchars($event["title"]); ?>
    <form method="post" action="register.php">
      <input type="hidden" name="event_id" value="<?php echo
        $event["id"]; ?>">
      <input name="student_name" required>
      <input name="email" type="email" required>
      <button>Register</button>
    </form>
  </article>
<?php endforeach; ?>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Etkinlik Kayıt Sistemi: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
article { border: 1px solid #ccc; padding: 12px;
margin:
  10px 0; }
input, button { padding: 8px; margin: 4px; }
.seats { font-weight: bold; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

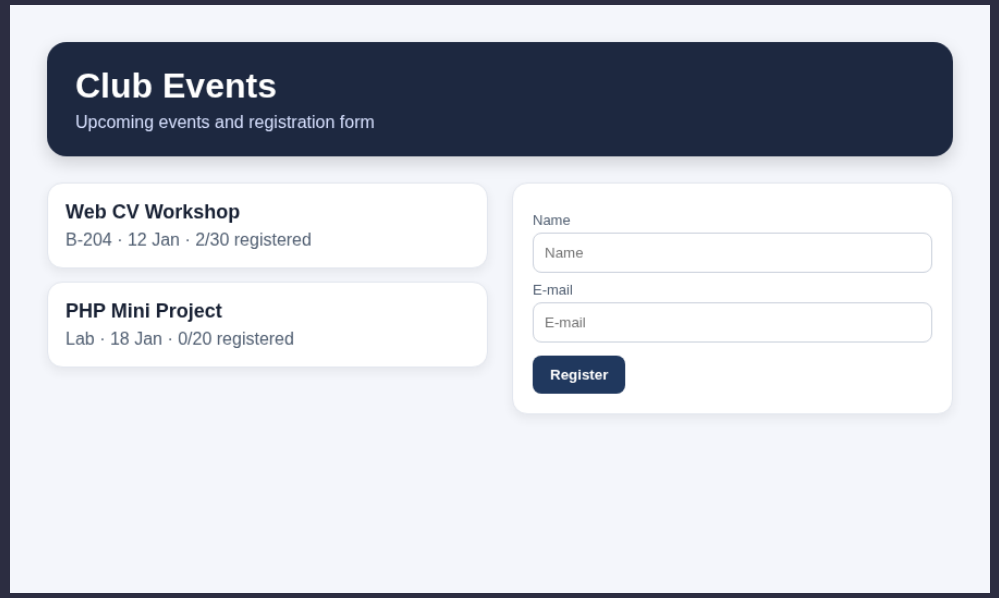
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Etkinlik Kayıt Sistemi: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
const form =
  document.querySelector("form[action='register.php']");
form.addEventListener("submit", function (e) {
  const email = form.email.value.trim();
  if (!email.includes("@")) {
    e.preventDefault();
    alert("Please enter a valid e-mail address.");
  }
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Etkinlik Kayıt Sistemi: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$sql = "SELECT e.id, e.title, e.capacity,
        COUNT(r.id) AS registered
        FROM events e
        LEFT JOIN registrations r ON r.event_id = e.id
        GROUP BY e.id
        ORDER BY e.event_date";
$stmt = $pdo->query($sql);
while ($row = $stmt->fetch()) {
  echo htmlspecialchars($row["title"]);
  echo " - " . $row["registered"] . "/" . $row["capacity"];
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Etkinlik Kayıt Sistemi: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
if ($_SERVER["REQUEST_METHOD"] === "POST") {
  $name = trim($_POST["student_name"] ?? "");
  $email = trim($_POST["email"] ?? "");
  $eventId = (int)($_POST["event_id"] ?? 0);

  $stmt = $pdo->prepare(
    "INSERT INTO registrations (event_id, student_name, email)
     VALUES (?, ?, ?)"
  );
  $stmt->execute([$eventId, $name, $email]);
  header("Location: index.php?registered=1");
}
```

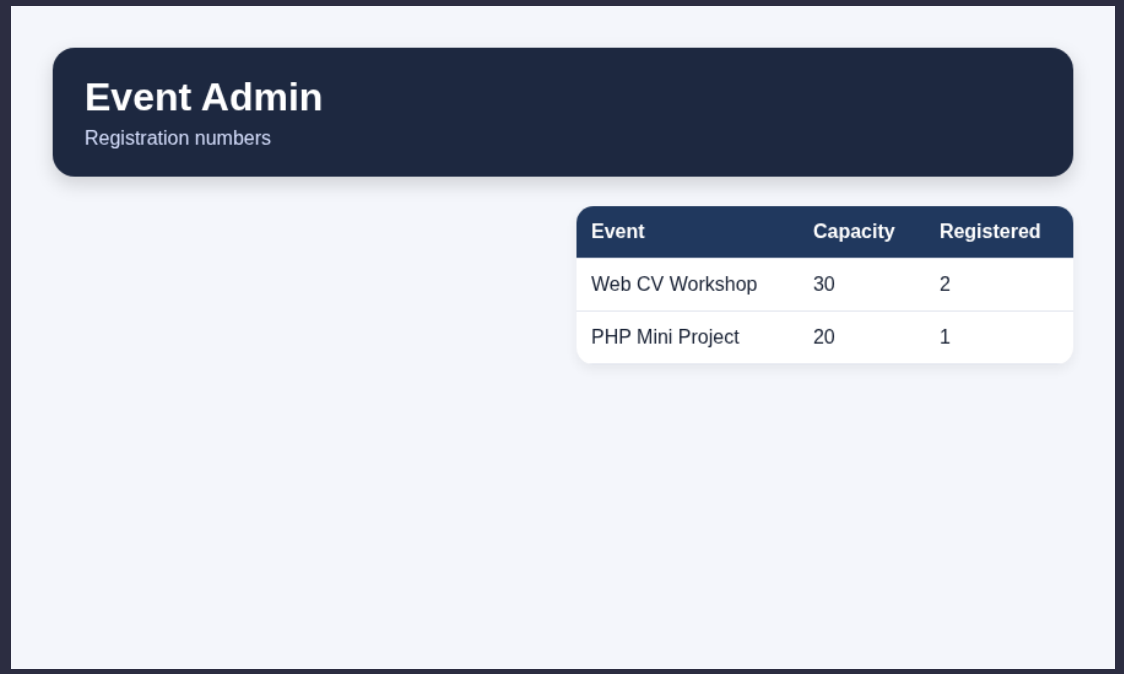
DB ÖNCESİ / SONRASI

```
BEFORE registrations
id | event_id | student_name
1  | 1        | Ada Yılmaz
2  | 1        | Mert Kaya

POST: event_id=2, student_name=Ece, email=ece@mail.com

AFTER registrations
id | event_id | student_name
1  | 1        | Ada Yılmaz
2  | 1        | Mert Kaya
3  | 2        | Ece Demir
```

Etkinlik Kayıt Sistemi: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Etkinlik dolduysa butonu pasifleştir
- Aynı e-posta aynı etkinliğe ikinci kez kayıt olamasın
- Admin için CSV dışa aktarma ekle

Mini Site 2: Randevu Takip Sistemi

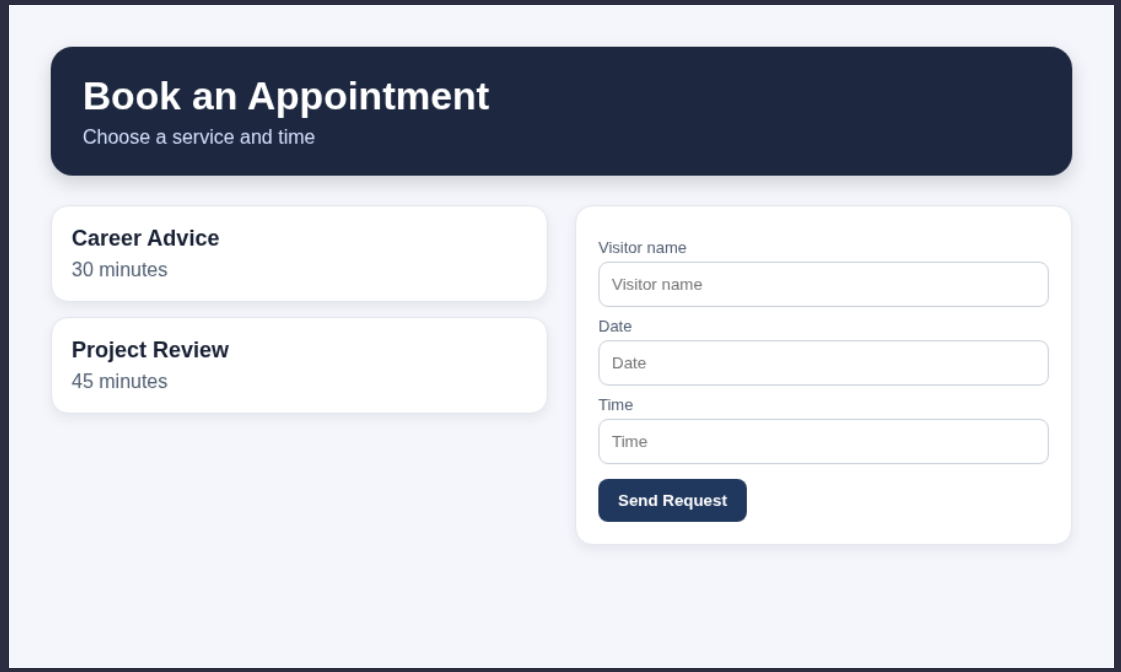
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Küçük bir danışmanlık ofisi öğrencilerden randevu talebi alır. Öğrenci servis ve saat seçer; görevli randevuları onaylandı/bekliyor olarak yönetir.

Öğrenci bu mini sitede neyi birleştirir?

- SELECT ile uygun saatleri listeleme
- POST ile randevu oluşturma
- UPDATE ile durum değiştirme
- JS ile tarih/saat kontrolü



Beklenen çıktı: ana ekran / kullanıcı formu

Randevu Takip Sistemi: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE services (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(80), duration_min INT
);
CREATE TABLE appointments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  service_id INT, visitor_name VARCHAR(80),
  appointment_date DATE, appointment_time TIME,
  status VARCHAR(20) DEFAULT 'pending',
  FOREIGN KEY (service_id) REFERENCES services(id)
);
```

DB ÖRNEK VERİ

```
services
id | name              | duration_min
1  | Career Advice     | 30
2  | Project Review    | 45

appointments
id | service_id | visitor_name | date       | time  | status
1  | 1          | Zeynep       | 2026-01-13 | 10:00 | pending
2  | 2          | Baran        | 2026-01-13 | 11:00 | approved
```

Randevu Takip Sistemi: Dosya akışı

Dosyalar ve görevleri

- index.php: servis ve tarih seçimi
- book.php: randevu kaydı
- admin.php: bekleyen randevular
- status.php: durum güncelleme

REQUEST AKIŞI

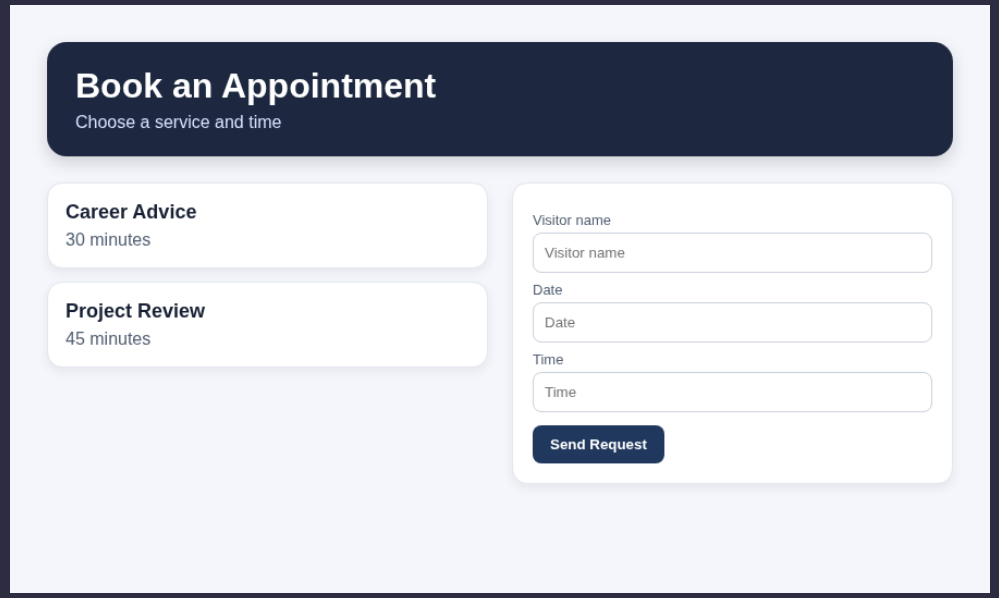
```
GET /index.php -> services tablosundan
seçenekleri çeker
POST /book.php -> appointments INSERT
POST /approve.php -> appointments.status UPDATE
GET /admin.php?status=pending -> filtrelenmiş
yönetim
  listesi
```

Randevu Takip Sistemi: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<?php include "db.php"; ?>
<h1>Book an Appointment</h1>
<form method="post" action="book.php" id="bookingForm">
  <input name="visitor_name" placeholder="Your name" required>
  <select name="service_id">
    <?php foreach ($services as $service): ?>
      <option value="<?php echo $service["id"]; ?>">
        <?php echo htmlspecialchars($service["name"]); ?>
      </option>
    <?php endforeach; ?>
  </select>
  <input name="appointment_date" type="date" required>
  <input name="appointment_time" type="time" required>
  <button>Send Request</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Randevu Takip Sistemi: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
form { max-width: 420px; }
input, select, button { display: block; margin:
8px 0;
  padding: 8px; }
table { border-collapse: collapse; }
td, th { border: 1px solid #ccc; padding: 6px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

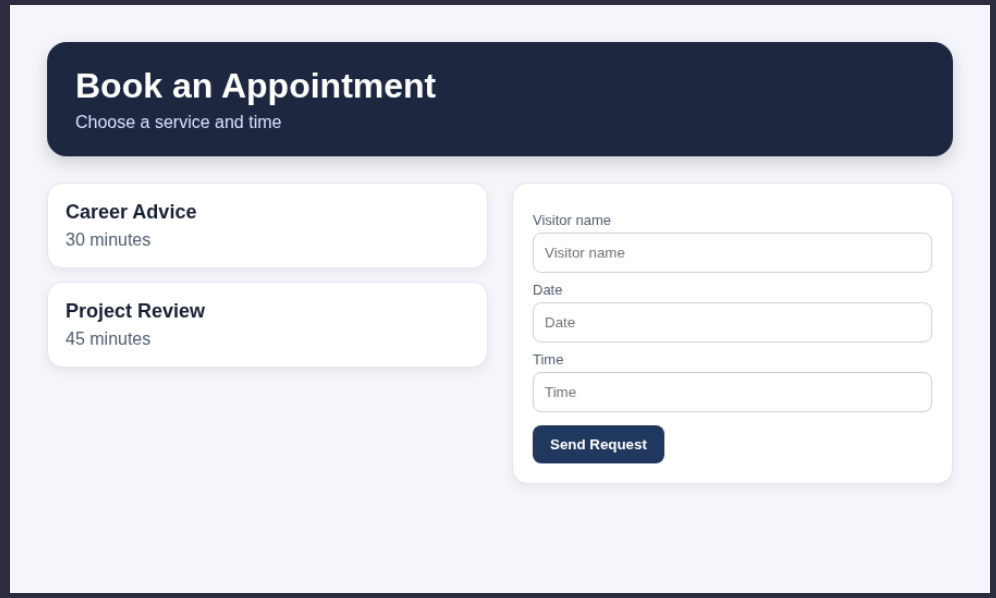
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Randevu Takip Sistemi: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelector("#bookingForm").addEventListener("submit",
  function(e) {
  const selectedDate = new Date(this.appointment_date.value);
  const today = new Date();
  today.setHours(0, 0, 0, 0);
  if (selectedDate < today) {
    e.preventDefault();
    alert("Please select today or a future date.");
  }
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Randevu Takip Sistemi: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$stmt = $pdo->query(
  "SELECT a.id, s.name AS service, a.visitor_name,
          a.appointment_date, a.appointment_time, a.status
   FROM appointments a
   JOIN services s ON s.id = a.service_id
   ORDER BY a.appointment_date, a.appointment_time"
);
foreach ($stmt as $row) {
  echo htmlspecialchars($row["visitor_name"]);
  echo " - " . htmlspecialchars($row["service"]);
  echo " (" . $row["status"] . ")";
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Randevu Takip Sistemi: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO appointments
   (service_id, visitor_name, appointment_date,
     appointment_time)
   VALUES (?, ?, ?, ?)"
);
$stmt->execute([
  $_POST["service_id"], trim($_POST["visitor_name"]),
  $_POST["appointment_date"], $_POST["appointment_time"]
]);

// approve.php
$update = $pdo->prepare("UPDATE appointments SET
  status='approved' WHERE id=?");
$update->execute([(int)$_POST["id"]]);
```

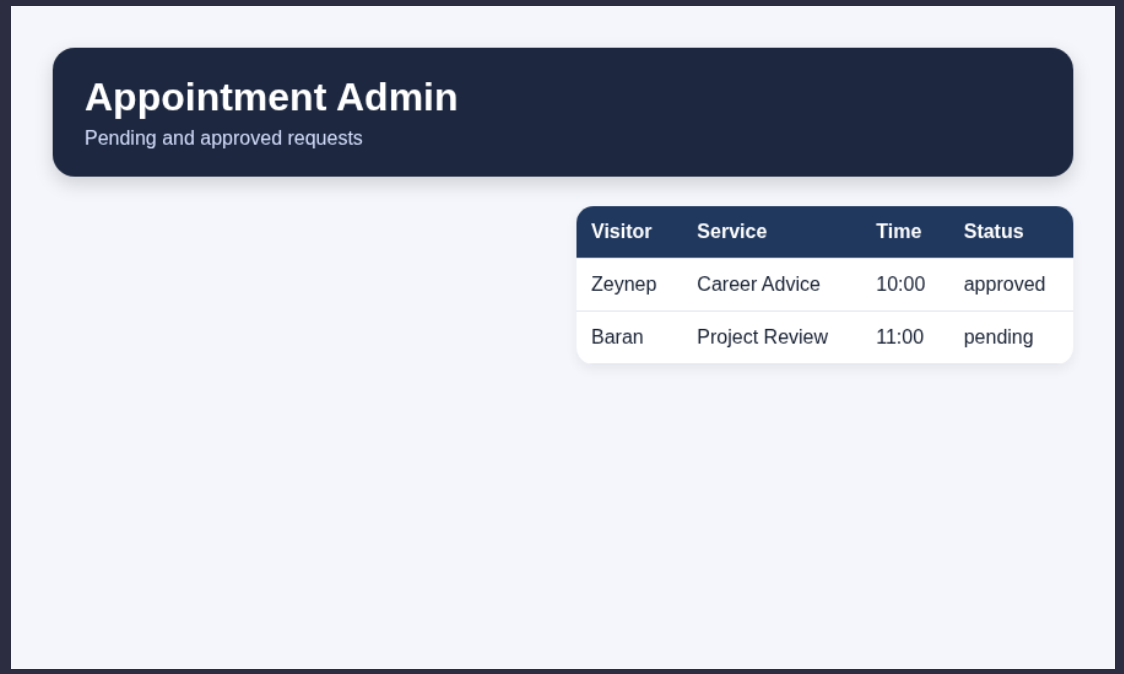
DB ÖNCESİ / SONRASI

```
BEFORE appointments
id | visitor | date       | time  | status
1  | Zeynep  | 2026-01-13 | 10:00 | pending

POST approve: id=1

AFTER appointments
id | visitor | date       | time  | status
1  | Zeynep  | 2026-01-13 | 10:00 | approved
```

Randevu Takip Sistemi: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:

1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Aynı saat için çakışan randevu engeli
- Sadece pending kayıtları admin panelinde göster
- Randevu iptal butonu ekle

Mini Site 3: Destek Talep Paneli

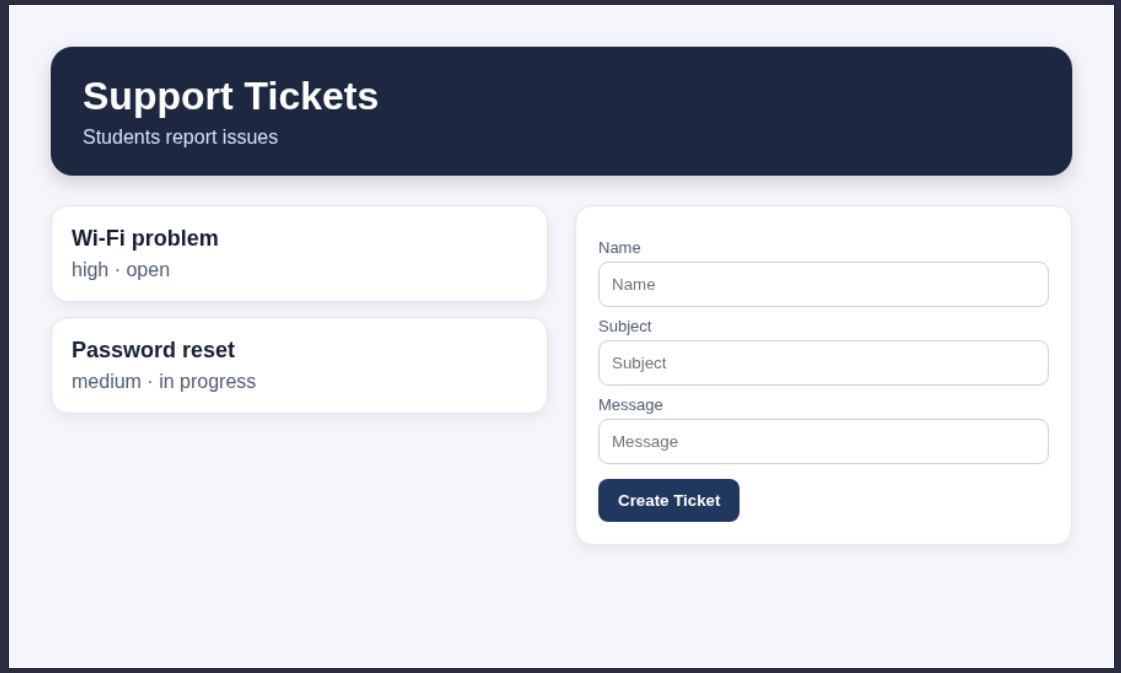
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Bir okul teknik destek masası öğrencilerden talep toplar. Öğrenci sorununu yazar; görevli durum ve öncelik değerlerini günceller.

Öğrenci bu mini sitede neyi birleştirir?

- Textarea ile açıklama alma
- GET ile durum filtresi
- UPDATE ile status değişimi
- COUNT/GROUP BY ile özet rapor



Beklenen çıktı: ana ekran / kullanıcı formu

Destek Talep Paneli: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE tickets (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_name VARCHAR(80), subject VARCHAR(120),
  message TEXT, priority VARCHAR(20), status VARCHAR(20)
    DEFAULT 'open',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

DB ÖRNEK VERİ

```
tickets
id | student_name | subject         | priority | status
1  | Deniz        | Wi-Fi problem   | high     | open
2  | Elif         | Password reset  | medium   | in_progress
3  | Can          | Printer issue   | low      | closed
```

Destek Talep Paneli: Dosya akışı

Dosyalar ve görevleri

- tickets.php: talepleri listeler
- new-ticket.php: talep formu
- save-ticket.php: INSERT
- update-ticket.php: status UPDATE

REQUEST AKIŞI

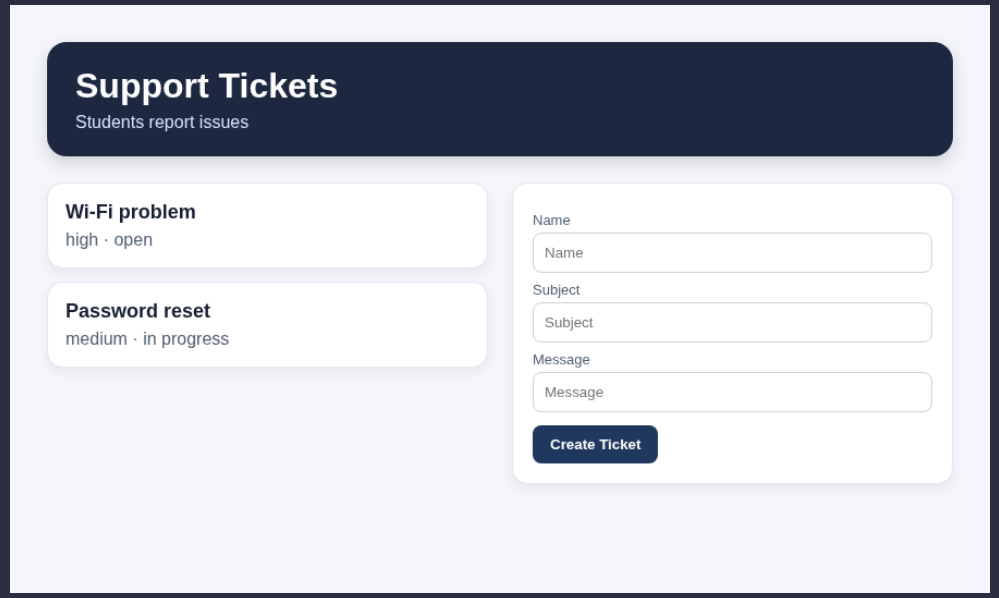
```
GET /tickets.php?status=open -> sadece açık
talepler
POST /save-ticket.php -> yeni talep INSERT
POST /update-ticket.php -> status UPDATE
GET /report.php -> status bazlı GROUP BY raporu
```

Destek Talep Paneli: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>New Support Ticket</h1>
<form method="post" action="save-ticket.php" id="ticketForm">
  <input name="student_name" placeholder="Name" required>
  <input name="subject" placeholder="Subject" required>
  <select name="priority">
    <option>low</option><option>medium</option><option>high</option>
  </select>
  <textarea name="message" required></textarea>
  <button>Create Ticket</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Destek Talep Paneli: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
textarea { width: 360px; height: 90px; }
.open { font-weight: bold; }
.closed { text-decoration: line-through; }
input, select, textarea, button { margin: 6px 0;
  display: block; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

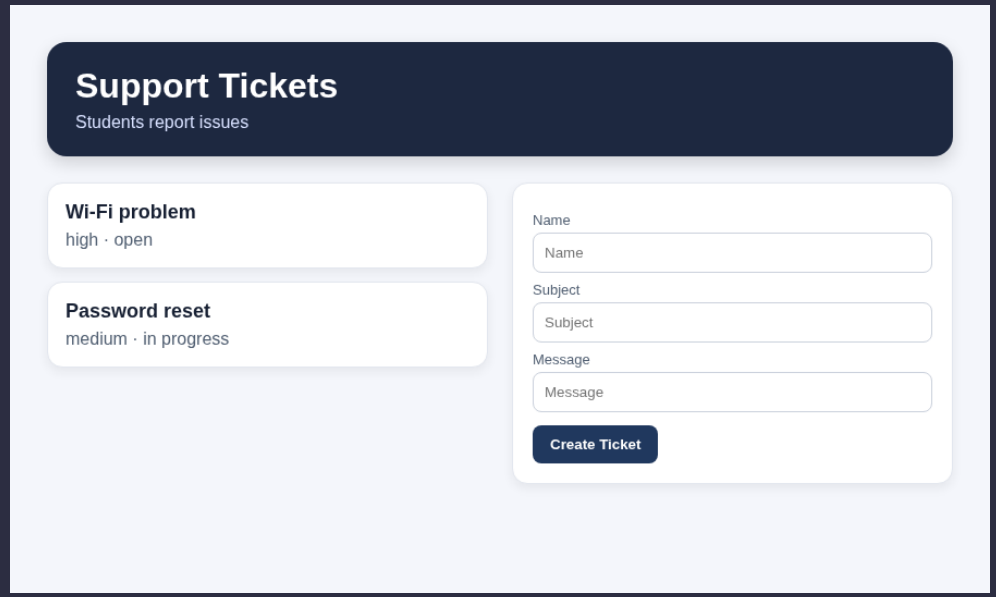
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Destek Talep Paneli: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelector("#ticketForm").addEventListener("submit",
  function(e) {
  const message = this.message.value.trim();
  if (message.length < 15) {
    e.preventDefault();
    alert("Please describe the problem with at least 15
      characters.");
  }
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Destek Talep Paneli: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$status = $_GET["status"] ?? "open";
$stmt = $pdo->prepare(
  "SELECT * FROM tickets WHERE status = ? ORDER BY created_at
    DESC"
);
$stmt->execute([$status]);
foreach ($stmt as $ticket) {
  echo htmlspecialchars($ticket["subject"]);
  echo " - " . htmlspecialchars($ticket["priority"]);
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Destek Talep Paneli: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO tickets (student_name, subject, message,
    priority)
   VALUES (?, ?, ?, ?)"
);
$stmt->execute([
  trim($_POST["student_name"]), trim($_POST["subject"]),
  trim($_POST["message"]), $_POST["priority"]
]);

$upd = $pdo->prepare("UPDATE tickets SET status=? WHERE id=?");
$upd->execute([$_POST["status"], (int)$_POST["id"]]);
```

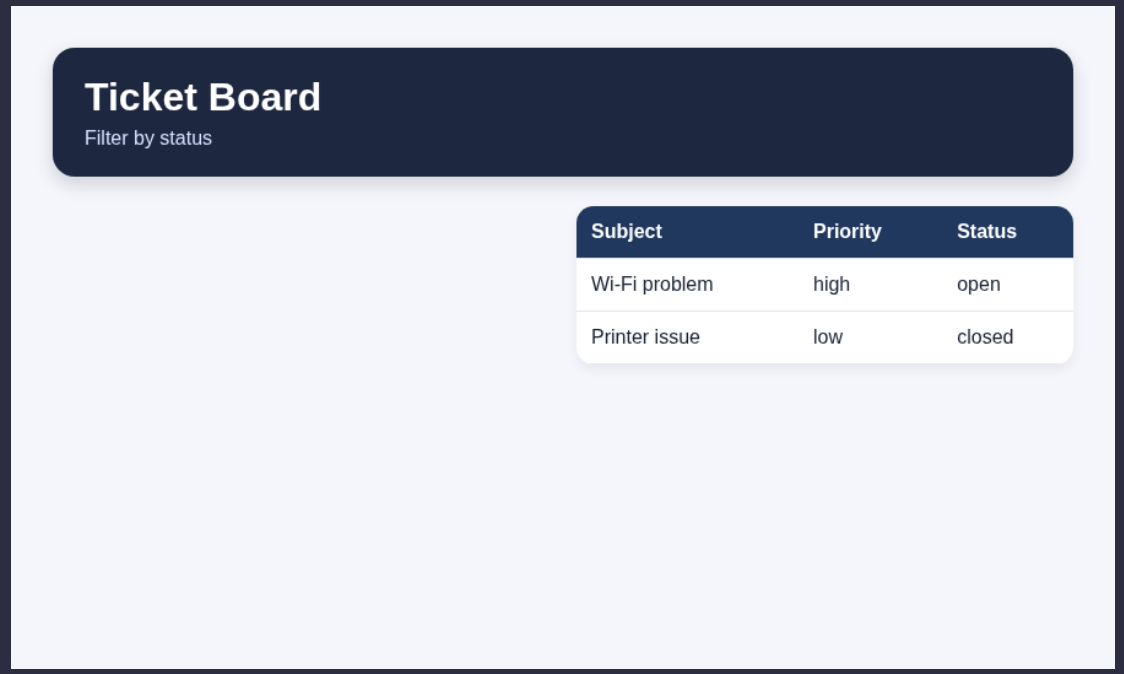
DB ÖNCESİ / SONRASI

```
BEFORE tickets
id | subject       | status
1  | Wi-Fi problem | open

POST update: id=1, status=in_progress

AFTER tickets
id | subject       | status
1  | Wi-Fi problem | in_progress
```

Destek Talep Paneli: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:

1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- High priority talepleri en üstte göster
- Kapalı talepler için kapatma tarihi ekle
- Admin notu alanı ekle

Mini Site 4: Ürün Stok ve Sepet Önizleme

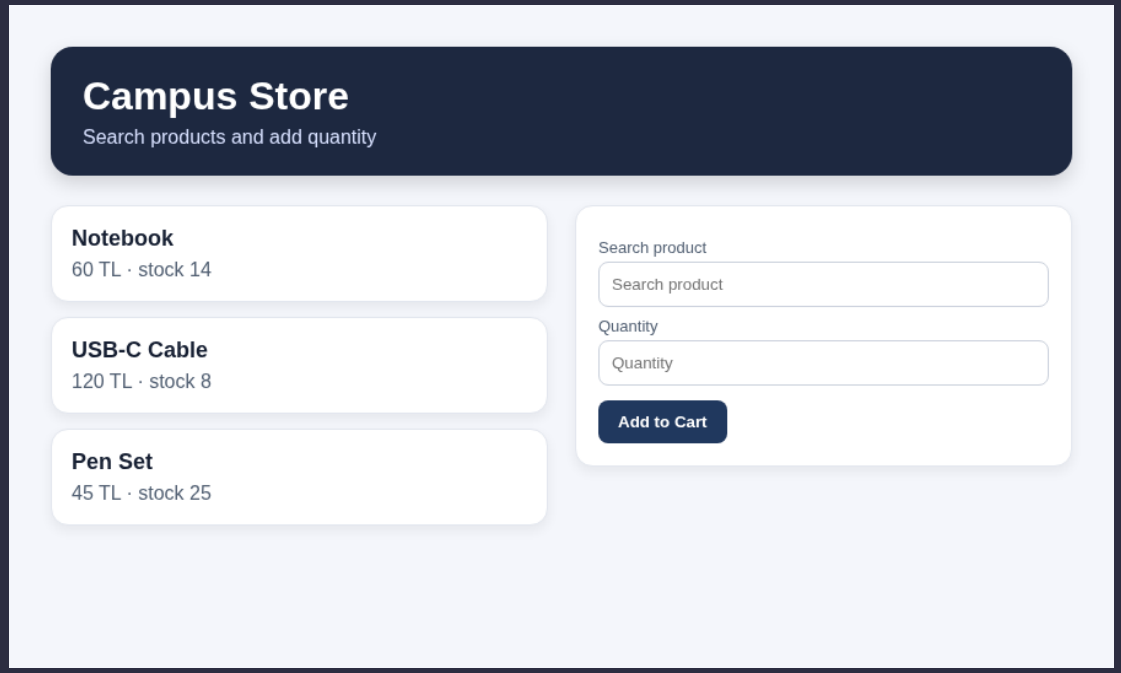
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Küçük bir kırtasiye sitesi ürünleri listeler. Ziyaretçi arama yapar, miktar girer; sistem stok yeterli mi kontrol eder ve sepet önizlemesi gösterir.

Öğrenci bu mini sitede neyi birleştirir?

- WHERE ile ürün arama
- JS ile toplam fiyat hesaplama
- POST ile sepet satırı ekleme
- UPDATE ile stok azaltma



Beklenen çıktı: ana ekran / kullanıcı formu

Ürün Stok ve Sepet Önizleme: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE products (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100), price DECIMAL(8,2), stock INT
);
CREATE TABLE cart_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  product_id INT, quantity INT,
  FOREIGN KEY (product_id) REFERENCES products(id)
);
```

DB ÖRNEK VERİ

```
products
id | name           | price | stock
1  | Notebook       | 60.00 | 14
2  | USB-C Cable    | 120.00| 8
3  | Pen Set        | 45.00 | 25

cart_items
id | product_id | quantity
1  | 2          | 1
```

Ürün Stok ve Sepet Önizleme: Dosya akışı

Dosyalar ve görevleri

- products.php: ürün listesi ve arama
- cart.php: sepet önizleme
- add-cart.php: stok kontrolü + INSERT
- checkout.php: stok UPDATE

REQUEST AKIŞI

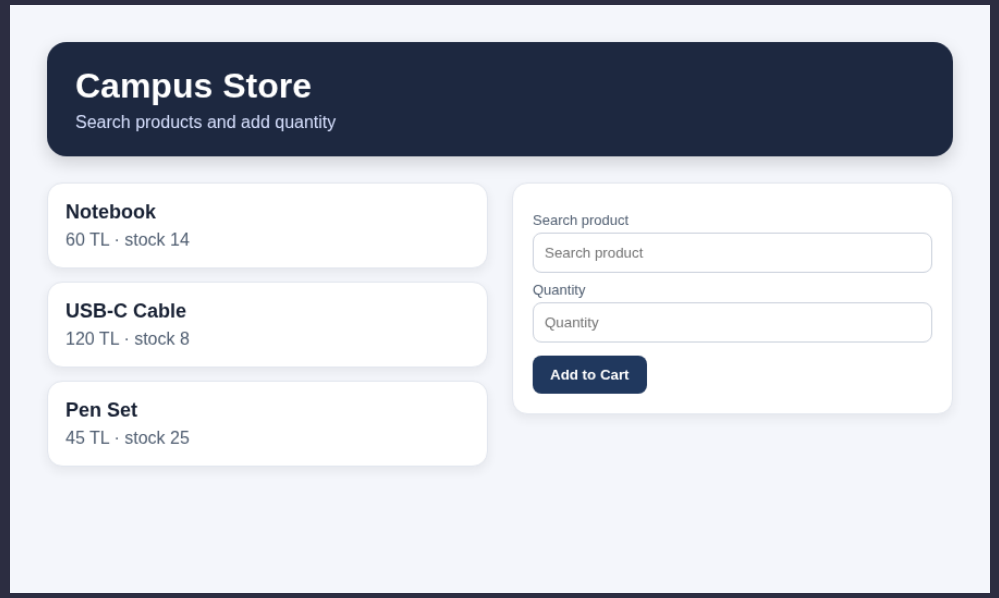
```
GET /products.php?q=pen -> WHERE name LIKE
POST /add-cart.php -> cart_items INSERT
POST /checkout.php -> products.stock UPDATE
GET /cart.php -> JOIN products + cart_items
```

Ürün Stok ve Sepet Önizleme: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<?php foreach ($products as $product): ?>
  <form method="post" action="add-cart.php" class="product">
    <h3><?php echo htmlspecialchars($product["name"]); ?></h3>
    <p><?php echo $product["price"]; ?> TL</p>
    <input type="hidden" name="product_id" value="<?php echo
      $product["id"]; ?>">
    <input name="quantity" type="number" min="1" value="1">
    <button>Add to Cart</button>
  </form>
<?php endforeach; ?>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Ürün Stok ve Sepet Önizleme: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
.product { border: 1px solid #ccc; margin: 8px;
padding:
  10px; }
.price { font-weight: bold; }
.warning { color: #b00020; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

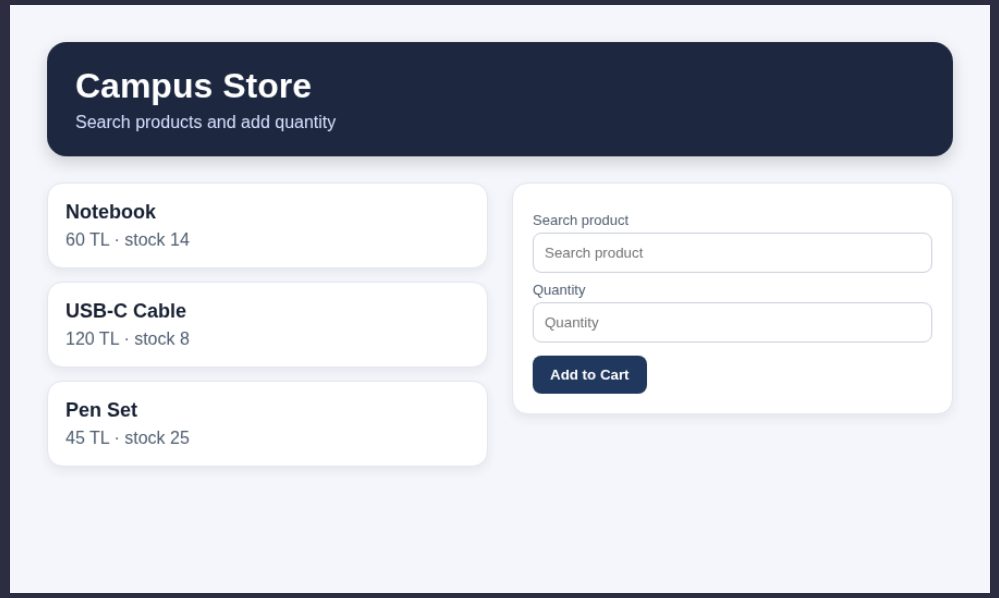
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Ürün Stok ve Sepet Önizleme: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelectorAll(".product").forEach(function(form) {
  form.quantity.addEventListener("input", function() {
    const price = Number(form.dataset.price || 0);
    const total = price * Number(this.value || 0);
    form.querySelector(".total").textContent = total.toFixed(2)
      + " TL";
  });
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Ürün Stok ve Sepet Önizleme: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```php
$q = $_GET["q"] ?? "";
$stmt = $pdo->prepare(
  "SELECT * FROM products
   WHERE name LIKE ?
   ORDER BY name"
);
$stmt->execute(["%" . $q . "%"]);
foreach ($stmt as $product) {
  echo htmlspecialchars($product["name"]);
  echo " - Stock: " . $product["stock"];
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Ürün Stok ve Sepet Önizleme: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$productId = (int)$_POST["product_id"];
$qty = (int)$_POST["quantity"];

$check = $pdo->prepare("SELECT stock FROM products WHERE id=?");
$check->execute([$productId]);
$stock = (int)$check->fetchColumn();

if ($qty <= $stock) {
  $stmt = $pdo->prepare("INSERT INTO cart_items (product_id,
    quantity) VALUES (?, ?)");
  $stmt->execute([$productId, $qty]);
}
```

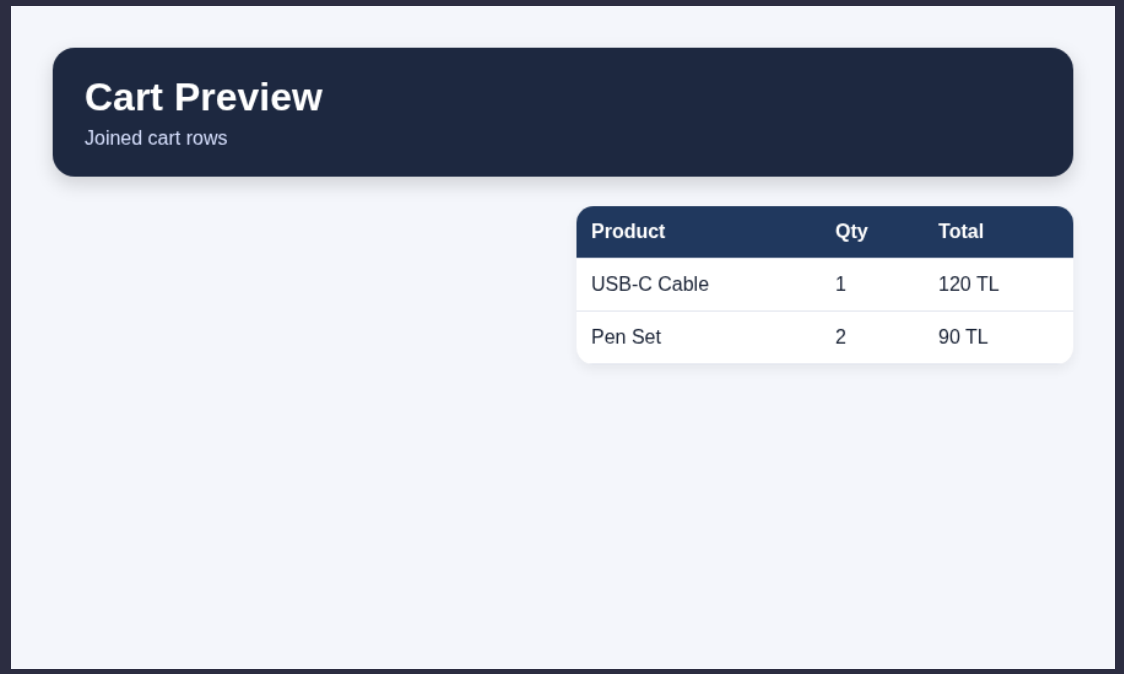
DB ÖNCESİ / SONRASI

```
BEFORE products
id | name         | stock
2  | USB-C Cable | 8

POST checkout: product_id=2, quantity=2

AFTER products
id | name         | stock
2  | USB-C Cable | 6
```

Ürün Stok ve Sepet Önizleme: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Stok yetersizse hata mesajı göster
- Sepetten ürün silme ekle
- Toplam tutar için SUM hesapla

Mini Site 5: Kütüphane Rezervasyon Sistemi

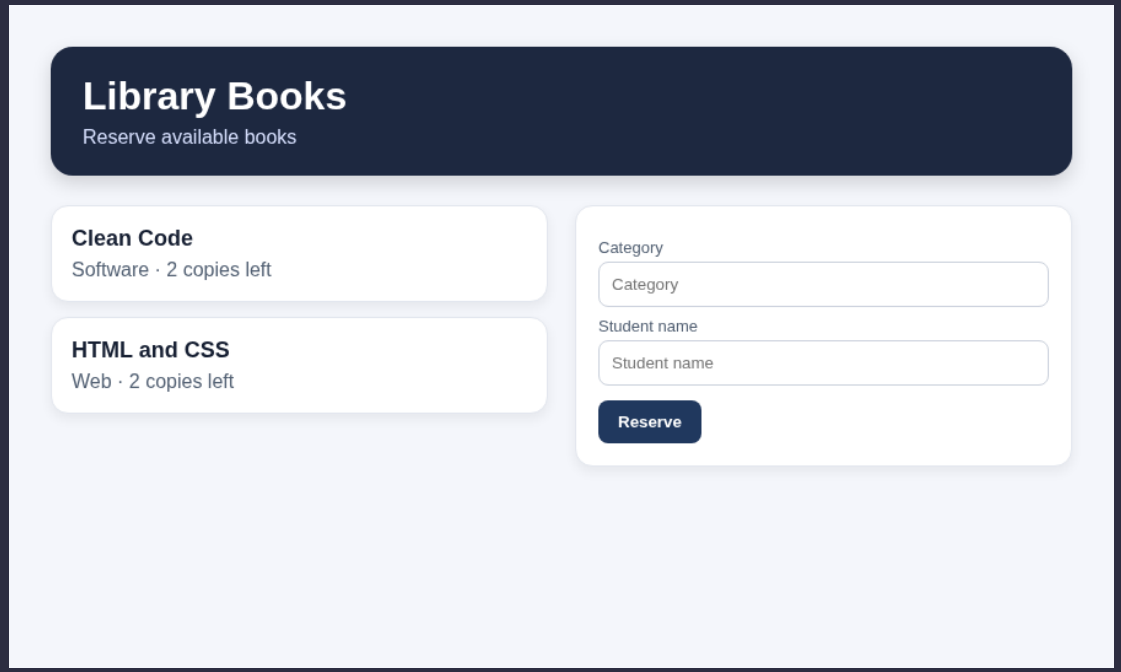
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Bir okul kütüphanesi kitapların uygunluk durumunu gösterir. Öğrenci kitap seçer, rezervasyon yapar; görevli teslim edildi/döndü durumlarını takip eder.

Öğrenci bu mini sitede neyi birleştirir?

- JOIN ile kitap + rezervasyon listesi
- POST ile rezervasyon oluşturma
- UPDATE ile durum yönetimi
- GET ile kategori arama



Beklenen çıktı: ana ekran / kullanıcı formu

Kütüphane Rezervasyon Sistemi: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE books (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(120), author VARCHAR(80), category
    VARCHAR(50), copies INT
);
CREATE TABLE reservations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  book_id INT, student_name VARCHAR(80), status VARCHAR(20)
    DEFAULT 'reserved',
  FOREIGN KEY (book_id) REFERENCES books(id)
);
```

DB ÖRNEK VERİ

```
books
id | title              | author        | category | copies
1  | Clean Code         | R. Martin     | Software | 3
2  | HTML and CSS       | J. Duckett    | Web      | 2

reservations
id | book_id | student_name | status
1  | 1       | Selin        | reserved
```

Kütüphane Rezervasyon Sistemi: Dosya akışı

Dosyalar ve görevleri

- books.php: kitap kataloğu
- reserve.php: rezervasyon kaydı
- my-reservations.php: öğrenci listesi
- return.php: durum UPDATE

REQUEST AKIŞI

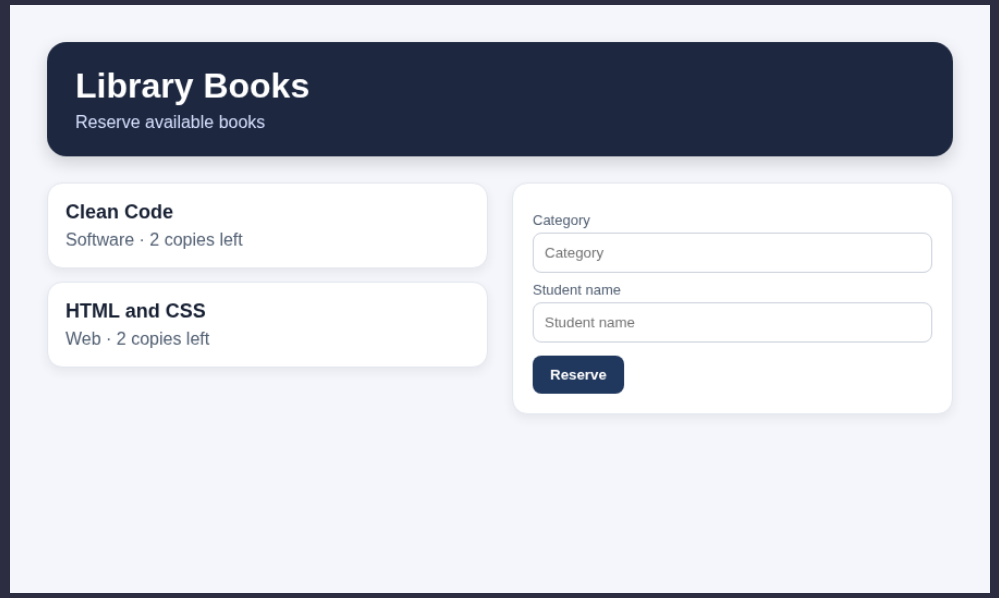
```
GET /books.php?category=Web -> kategori filtresi
POST /reserve.php -> reservations INSERT
POST /return.php -> status UPDATE
GET /my-reservations.php?name=Selin -> öğrenciye
göre
  liste
```

Kütüphane Rezervasyon Sistemi: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>Library Books</h1>
<form method="get" action="books.php">
  <input name="category" placeholder="Category">
  <button>Filter</button>
</form>
<?php foreach ($books as $book): ?>
  <section>
    <strong><?php echo htmlspecialchars($book["title"]);
      ?></strong>
    <form method="post" action="reserve.php">
      <input type="hidden" name="book_id" value="<?php echo
        $book["id"]; ?>">
      <input name="student_name" required>
      <button>Reserve</button>
    </form>
  </section>
<?php endforeach; ?>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Kütüphane Rezervasyon Sistemi: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
section { border-bottom: 1px solid #ccc; padding:
10px
  0; }
input, button { padding: 7px; margin: 4px; }
.available { color: #0a7a2f; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

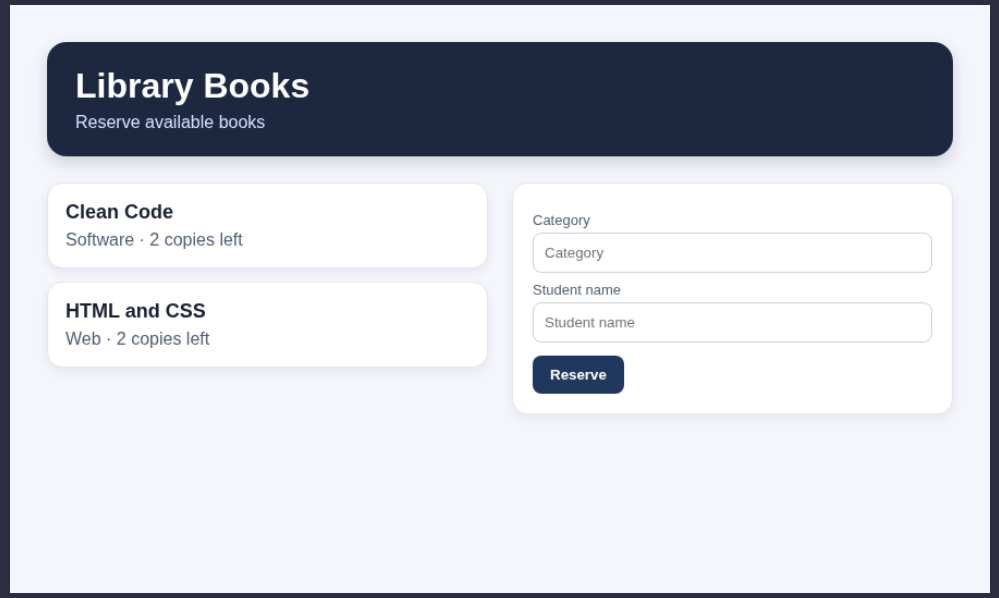
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Kütüphane Rezervasyon Sistemi: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelectorAll("form[action='reserve.php']").forEach(fu
nction(form)
  {
  form.addEventListener("submit", function(e) {
    if (!confirm("Do you want to reserve this book?")) {
      e.preventDefault();
    }
  });
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Kütüphane Rezervasyon Sistemi: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$category = $_GET["category"] ?? "";
$stmt = $pdo->prepare(
  "SELECT b.*, COUNT(r.id) AS reserved_count
   FROM books b
   LEFT JOIN reservations r ON r.book_id = b.id AND
     r.status='reserved'
   WHERE b.category LIKE ?
   GROUP BY b.id"
);
$stmt->execute(["%" . $category . "%"]);
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Kütüphane Rezervasyon Sistemi: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO reservations (book_id, student_name) VALUES (?,
    ?)"
);
$stmt->execute([
  (int)$_POST["book_id"], trim($_POST["student_name"])
]);

$return = $pdo->prepare("UPDATE reservations SET
  status='returned' WHERE id=?");
$return->execute([(int)$_POST["reservation_id"]]);
```

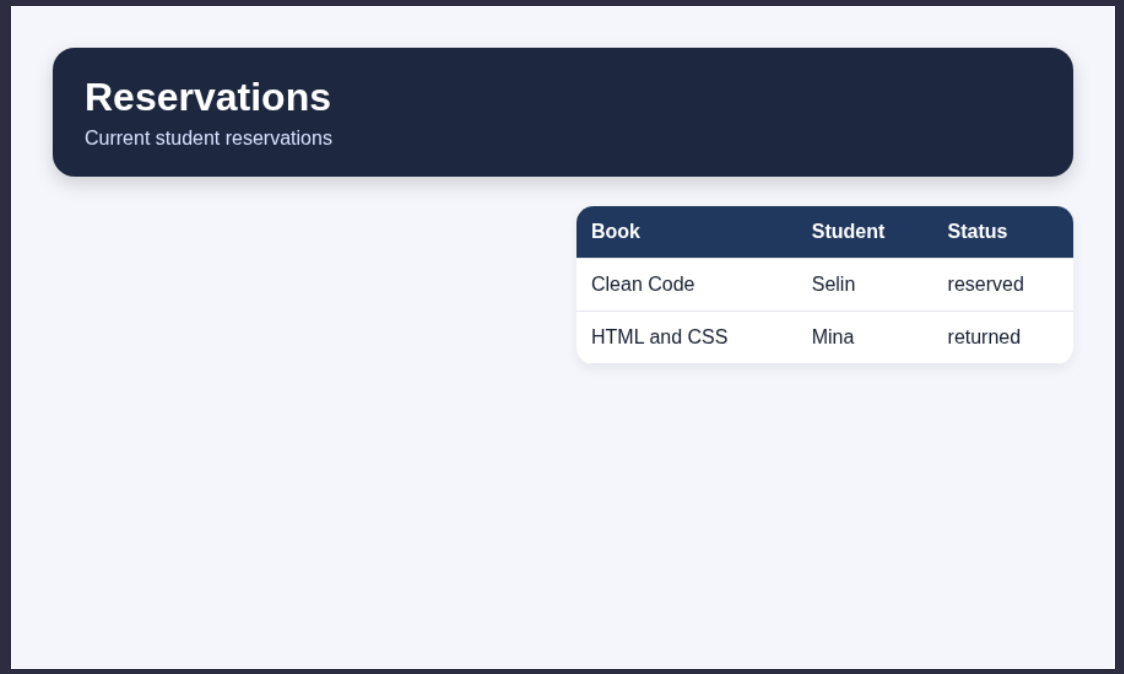
DB ÖNCESİ / SONRASI

```
BEFORE reservations
id | book_id | student | status
1  | 1       | Selin   | reserved

POST return: reservation_id=1

AFTER reservations
id | book_id | student | status
1  | 1       | Selin   | returned
```

Kütüphane Rezervasyon Sistemi: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:

1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Kopya sayısından fazla rezervasyonu engelle
- Yazar adına göre arama ekle
- Gecikmiş rezervasyon raporu oluştur

Mini Site 6: Restoran Menü ve Sipariş Taslağı

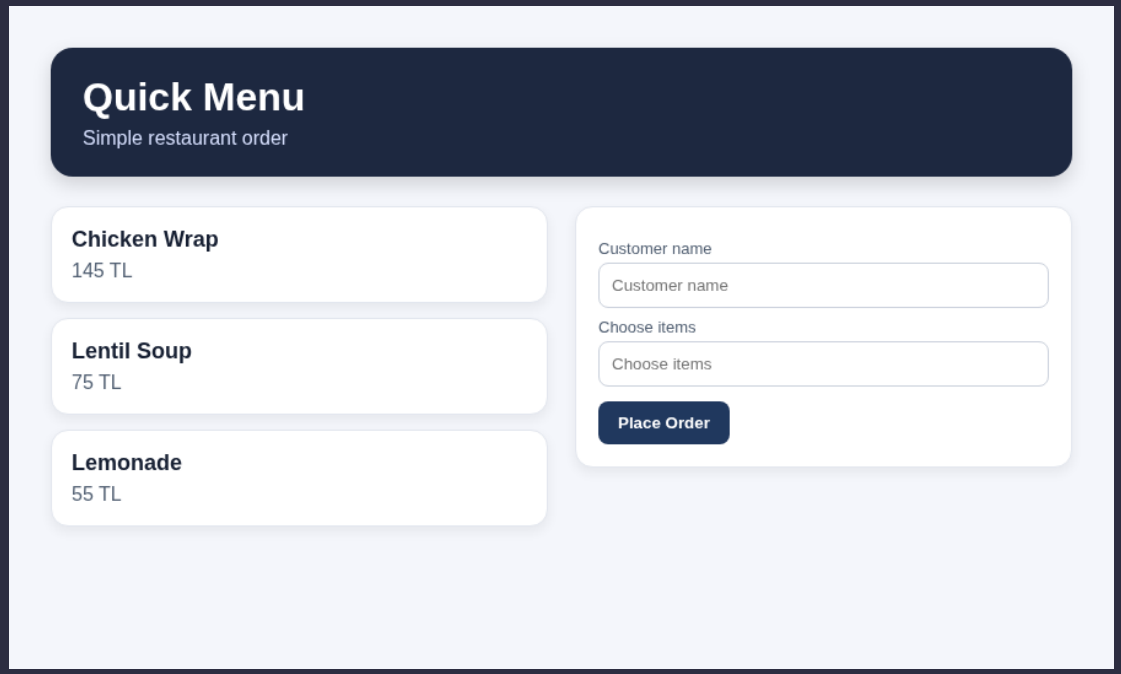
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Mahalle restoranı basit bir online menü yayınlar. Kullanıcı ürün ve adet seçer; sistem siparişi kaydeder ve mutfak ekranında bekleyen siparişleri gösterir.

Öğrenci bu mini sitede neyi birleştirir?

- Menü ürünlerini SELECT ile listeleme
- JS ile sipariş toplamı
- POST ile order + order_items ekleme
- DELETE ile iptal



Beklenen çıktı: ana ekran / kullanıcı formu

Restoran Menü ve Sipariş Taslağı: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE menu_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100), price DECIMAL(8,2), active TINYINT
    DEFAULT 1
);
CREATE TABLE orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  customer_name VARCHAR(80), status VARCHAR(20) DEFAULT
    'new'
);
CREATE TABLE order_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT, menu_item_id INT, quantity INT
);
```

DB ÖRNEK VERİ

```
menu_items
id | name          | price | active
1  | Chicken Wrap  | 145.00| 1
2  | Lentil Soup   | 75.00 | 1
3  | Lemonade      | 55.00 | 1

orders
id | customer_name | status
1  | Ayşe          | new
```

Restoran Menü ve Sipariş Taslağı: Dosya akışı

Dosyalar ve görevleri

- menu.php: ürün listesi ve sipariş formu
- place-order.php: INSERT işlemleri
- kitchen.php: bekleyen siparişler
- cancel-order.php: DELETE/UPDATE

REQUEST AKIŞI

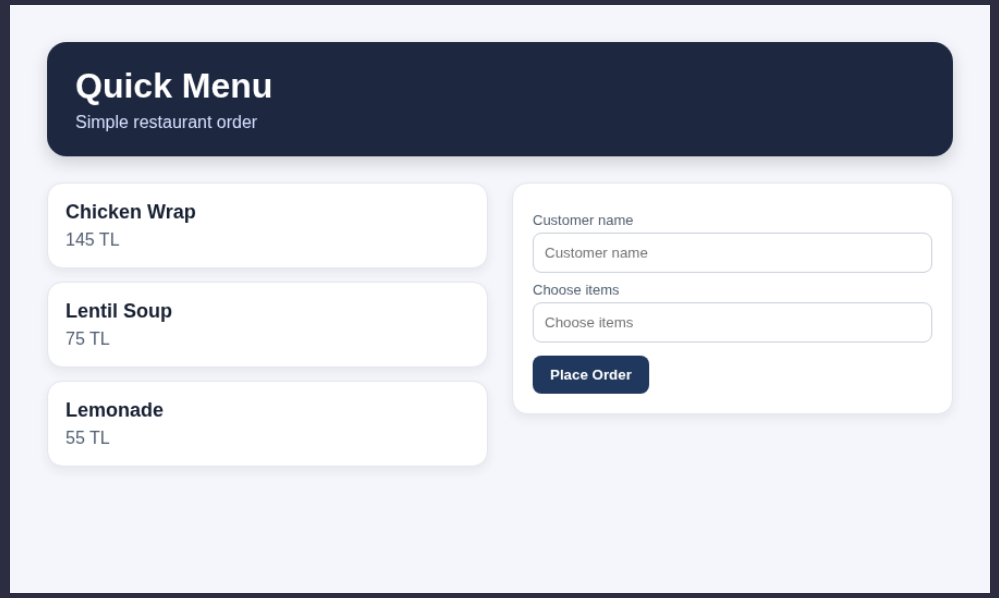
```
GET /menu.php -> aktif ürünler
POST /place-order.php -> orders + order_items
INSERT
GET /kitchen.php -> JOIN ile sipariş detayları
POST /cancel-order.php -> status='cancelled'
veya DELETE
```

Restoran Menü ve Sipariş Taslağı: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>Menu</h1>
<form method="post" action="place-order.php" id="orderForm">
  <input name="customer_name" placeholder="Your name" required>
  <?php foreach ($items as $item): ?>
    <label>
      <input type="checkbox" name="items[]" value="<?php echo
        $item["id"]; ?>">
      <?php echo htmlspecialchars($item["name"]); ?>
      <?php echo $item["price"]; ?> TL
    </label>
  <?php endforeach; ?>
  <button>Place Order</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Restoran Menü ve Sipariş Taslağı: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
label { display: block; margin: 8px 0; }
button { padding: 8px 12px; }
.total { font-weight: bold; margin-top: 12px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

Yapılmayacak şey

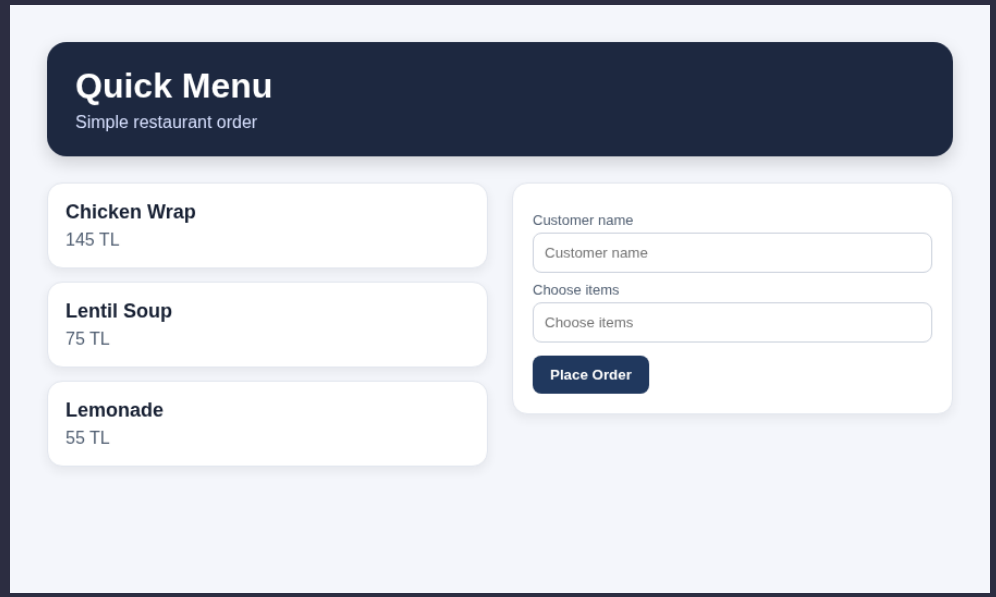
Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Restoran Menü ve Sipariş Taslağı: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
const prices = {1:145, 2:75, 3:55};
document.querySelectorAll("input[name='items[]']").forEach(functio
n(box)
  {
  box.addEventListener("change", function() {
    let total = 0;

document.querySelectorAll("input[name='items[]']:checked").forEach
(function(chosen)
      {
      total += prices[chosen.value];
    });
    document.querySelector("#total").textContent = total + "
      TL";
  });
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Restoran Menü ve Sipariş Taslağı: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$stmt = $pdo->query(
  "SELECT o.id, o.customer_name, m.name, oi.quantity
   FROM orders o
   JOIN order_items oi ON oi.order_id = o.id
   JOIN menu_items m ON m.id = oi.menu_item_id
   WHERE o.status = 'new'
   ORDER BY o.id DESC"
);
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Restoran Menü ve Sipariş Taslağı: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$pdo->beginTransaction();
$stmt = $pdo->prepare("INSERT INTO orders (customer_name) VALUES
  (?)");
$stmt->execute([trim($_POST["customer_name"])]);
$orderId = $pdo->lastInsertId();

$itemStmt = $pdo->prepare(
  "INSERT INTO order_items (order_id, menu_item_id, quantity)
    VALUES (?, ?, 1)"
);
foreach ($_POST["items"] as $itemId) {
  $itemStmt->execute([$orderId, (int)$itemId]);
}
$pdo->commit();
```

DB ÖNCESİ / SONRASI

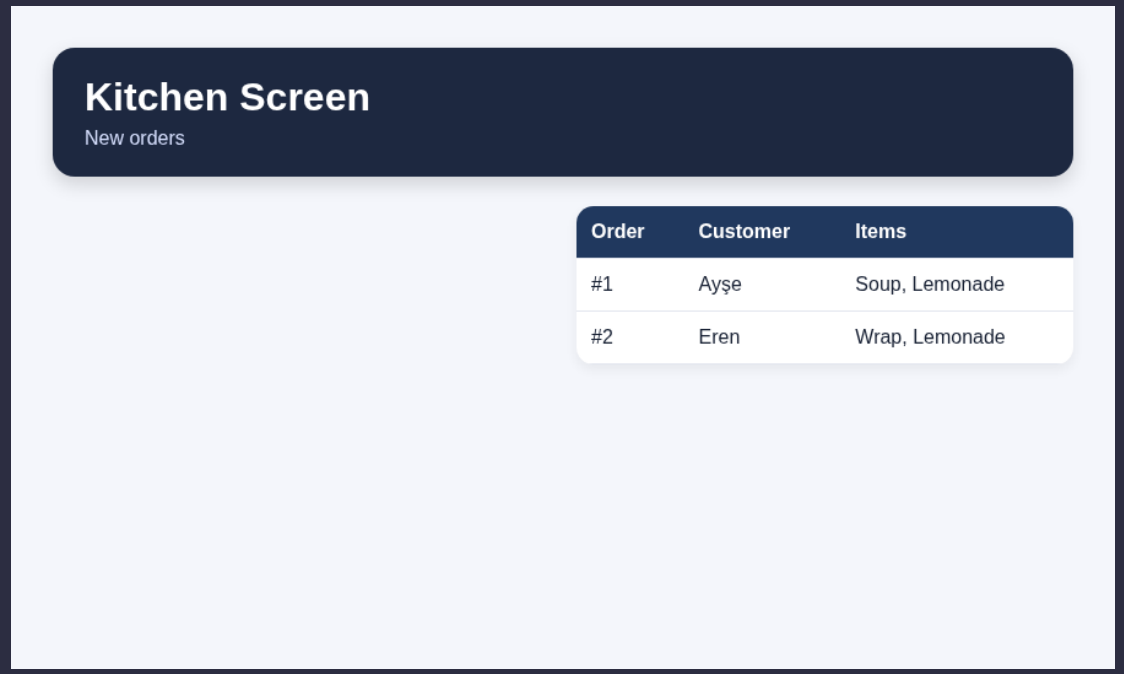
```
BEFORE orders
id | customer | status
1  | Ayşe     | new

POST order: customer=Eren, items=[1,3]

AFTER orders
id | customer | status
1  | Ayşe     | new
2  | Eren     | new

AFTER order_items
order_id | menu_item_id
2        | 1
2        | 3
```

Restoran Menü ve Sipariş Taslağı: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Adet seçimi ekle
- Sipariş toplamını SQL SUM ile hesapla
- Mutfak ekranında “hazır” butonu ekle

Mini Site 7: Gider Takip Uygulaması

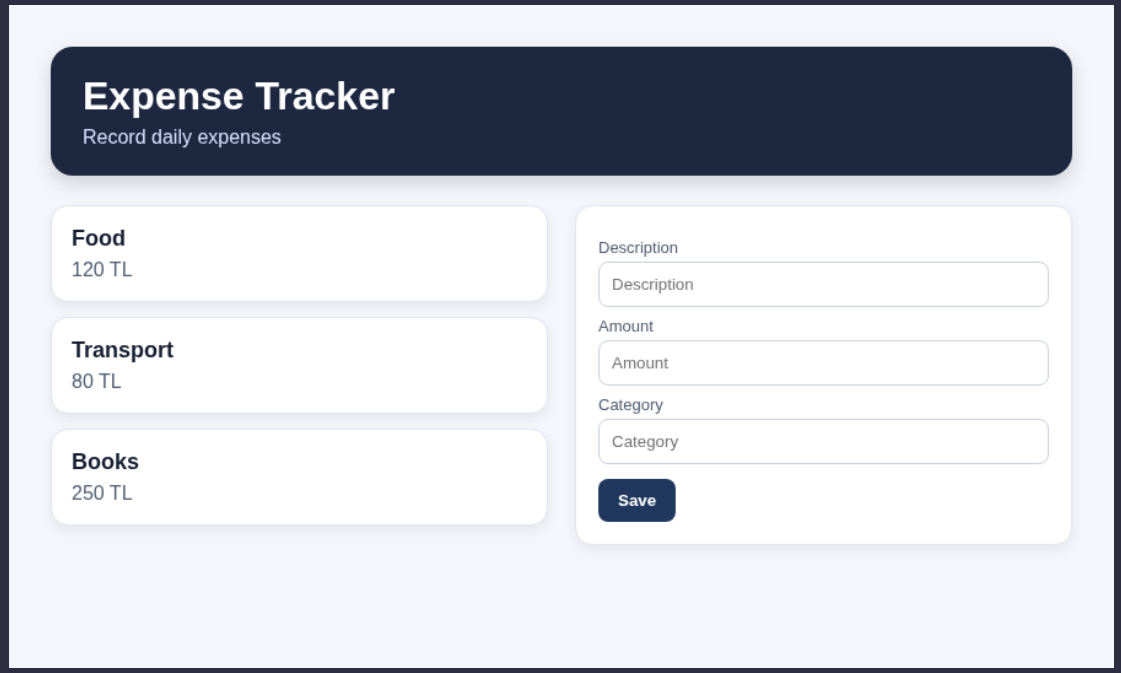
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Bir öğrenci aylık harcamalarını takip etmek ister. Kategori seçer, tutar ve açıklama girer; sistem toplamları ve kategori özetini gösterir.

Öğrenci bu mini sitede neyi birleştirir?

- DECIMAL veri tipi
- POST ile gider kaydı
- GROUP BY + SUM ile rapor
- GET ile ay filtresi



Beklenen çıktı: ana ekran / kullanıcı formu

Gider Takip Uygulaması: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE categories (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(60)
);
CREATE TABLE expenses (
  id INT AUTO_INCREMENT PRIMARY KEY,
  category_id INT, description VARCHAR(120), amount
    DECIMAL(8,2), expense_date DATE,
  FOREIGN KEY (category_id) REFERENCES categories(id)
);
```

DB ÖRNEK VERİ

```
categories
id | name
1  | Food
2  | Transport
3  | Books

expenses
id | category_id | description | amount | expense_date
1  | 1           | Lunch       | 120.00 | 2026-01-04
2  | 2           | Metro card  | 80.00  | 2026-01-05
```

Gider Takip Uygulaması: Dosya akışı

Dosyalar ve görevleri

- expenses.php: liste ve form
- save-expense.php: INSERT
- summary.php: kategori raporu
- delete-expense.php: silme işlemi

REQUEST AKIŞI

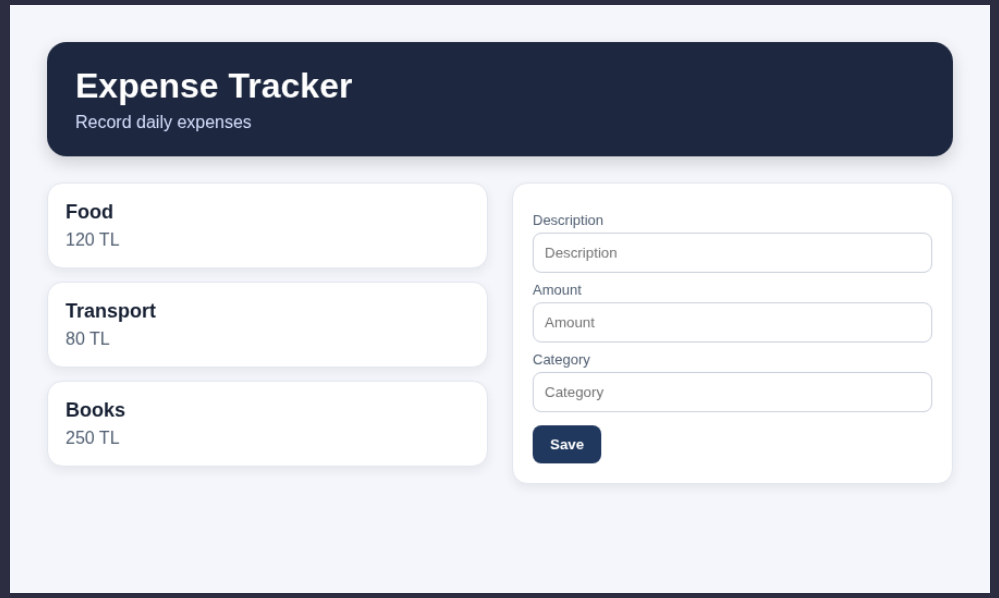
```
GET /expenses.php?month=2026-01 -> ay filtresi
POST /save-expense.php -> expenses INSERT
POST /delete-expense.php -> expenses DELETE
GET /summary.php -> GROUP BY + SUM raporu
```

Gider Takip Uygulaması: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>Expense Tracker</h1>
<form method="post" action="save-expense.php" id="expenseForm">
  <input name="description" placeholder="Description" required>
  <input name="amount" type="number" step="0.01" required>
  <input name="expense_date" type="date" required>
  <select name="category_id">
    <?php foreach ($categories as $cat): ?>
      <option value="<?php echo $cat["id"]; ?>">
        <?php echo htmlspecialchars($cat["name"]); ?>
      </option>
    <?php endforeach; ?>
  </select>
  <button>Save</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Gider Takip Uygulaması: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
input, select, button { margin: 6px 0; padding:
7px;
  display: block; }
table { border-collapse: collapse; }
th, td { border: 1px solid #ccc; padding: 6px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

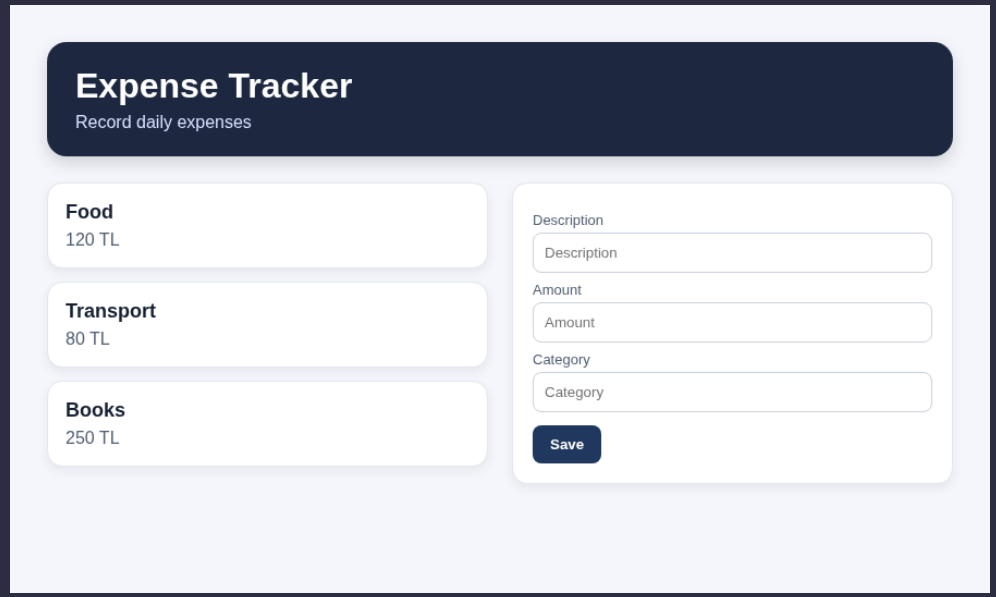
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Gider Takip Uygulaması: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelector("#expenseForm").addEventListener("submit",
  function(e) {
  const amount = Number(this.amount.value);
  if (amount <= 0) {
    e.preventDefault();
    alert("Amount must be greater than zero.");
  }
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Gider Takip Uygulaması: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$month = $_GET["month"] ?? "2026-01";
$stmt = $pdo->prepare(
  "SELECT c.name, SUM(e.amount) AS total
   FROM expenses e
   JOIN categories c ON c.id = e.category_id
   WHERE DATE_FORMAT(e.expense_date, '%Y-%m') = ?
   GROUP BY c.id"
);
$stmt->execute([$month]);
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Gider Takip Uygulaması: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO expenses (category_id, description, amount,
    expense_date)
   VALUES (?, ?, ?, ?)"
);
$stmt->execute([
  (int)$_POST["category_id"], trim($_POST["description"]),
  (float)$_POST["amount"], $_POST["expense_date"]
]);

$delete = $pdo->prepare("DELETE FROM expenses WHERE id=?");
$delete->execute([(int)$_POST["expense_id"]]);
```

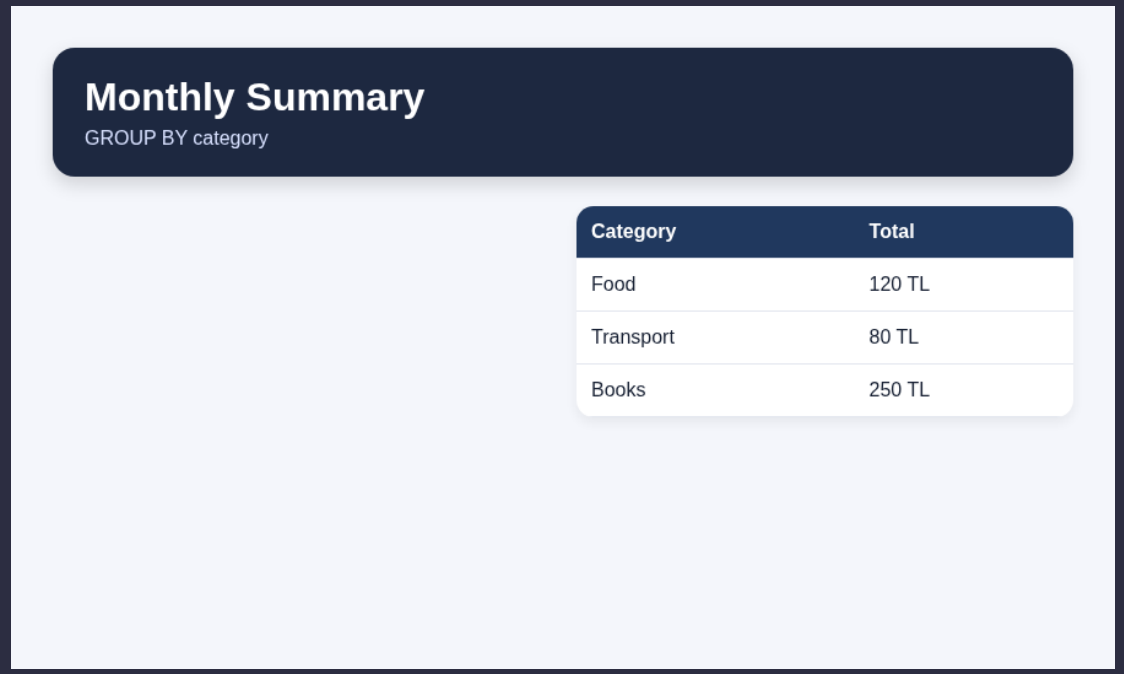
DB ÖNCESİ / SONRASI

```
BEFORE expenses
id | description | amount
1  | Lunch       | 120.00
2  | Metro card  | 80.00

POST delete: expense_id=2

AFTER expenses
id | description | amount
1  | Lunch       | 120.00
```

Gider Takip Uygulaması: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:

1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Bütçe limiti aşıldığında uyarı göster
- Kategoriye göre filtre ekle
- Aylık toplamı ana sayfada göster

Mini Site 8: Ders Geri Bildirim Formu

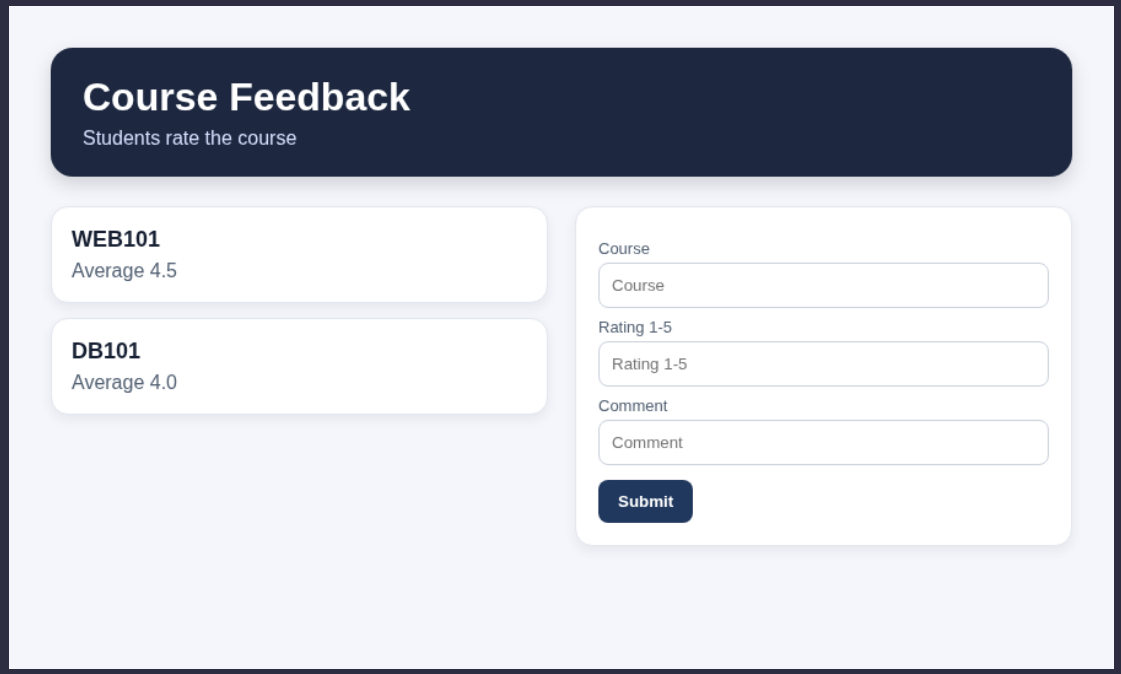
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Dönem sonunda öğrenciler ders hakkında kısa geri bildirim verir. Sistem puanları kaydeder, öğretim elemanı ortalama puanı ve yorumları görür.

Öğrenci bu mini sitede neyi birleştirir?

- Radio/select input kullanımı
- POST validation
- AVG ve COUNT ile rapor
- XSS'e karşı htmlspecialchars



Beklenen çıktı: ana ekran / kullanıcı formu

Ders Geri Bildirim Formu: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE courses (
  id INT AUTO_INCREMENT PRIMARY KEY,
  code VARCHAR(20), title VARCHAR(100)
);
CREATE TABLE feedback (
  id INT AUTO_INCREMENT PRIMARY KEY,
  course_id INT, rating INT, comment TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (course_id) REFERENCES courses(id)
);
```

DB ÖRNEK VERİ

```
courses
id | code    | title
1  | WEB101  | Web Programming
2  | DB101   | Database Basics

feedback
id | course_id | rating | comment
1  | 1         | 5      | Very useful
2  | 1         | 4      | More practice please
```

Ders Geri Bildirim Formu: Dosya akışı

Dosyalar ve görevleri

- feedback.php: form
- save-feedback.php: kayıt
- results.php: ortalama ve yorumlar
- db.php

REQUEST AKIŞI

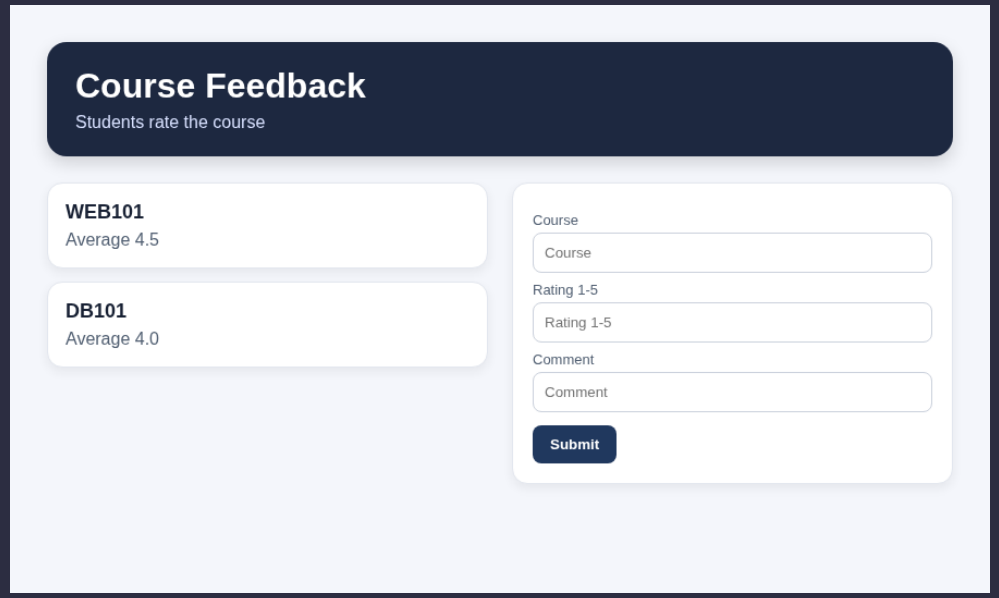
```
GET /feedback.php -> course seçeneklerini çeker
POST /save-feedback.php -> feedback INSERT
GET /results.php -> AVG + COUNT raporu
GET /results.php?course_id=1 -> detaylı yorum
listesi
```

Ders Geri Bildirim Formu: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>Course Feedback</h1>
<form method="post" action="save-feedback.php"
  id="feedbackForm">
  <select name="course_id">
    <?php foreach ($courses as $course): ?>
      <option value="<?php echo $course["id"]; ?>">
        <?php echo htmlspecialchars($course["code"]); ?>
      </option>
    <?php endforeach; ?>
  </select>
  <input name="rating" type="number" min="1" max="5" required>
  <textarea name="comment" required></textarea>
  <button>Submit</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Ders Geri Bildirim Formu: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
textarea { width: 360px; height: 80px; }
input, select, textarea, button { display: block;
  margin: 6px 0; padding: 7px; }
.rating { font-size: 22px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

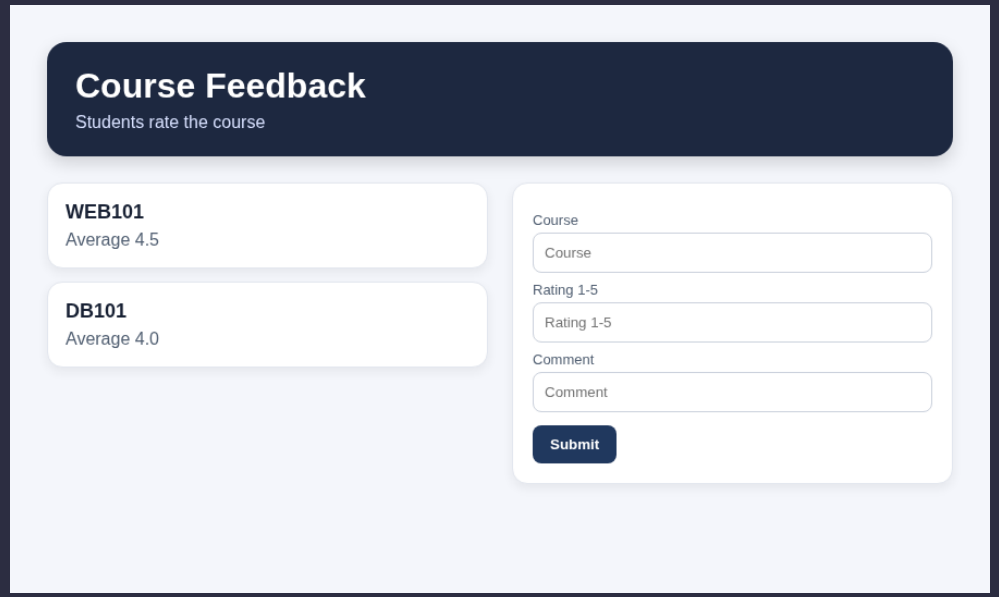
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Ders Geri Bildirim Formu: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelector("#feedbackForm").addEventListener("submit",
  function(e) {
  const rating = Number(this.rating.value);
  if (rating < 1 || rating > 5) {
    e.preventDefault();
    alert("Rating must be between 1 and 5.");
  }
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Ders Geri Bildirim Formu: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$stmt = $pdo->query(
  "SELECT c.code, COUNT(f.id) AS total_votes,
          ROUND(AVG(f.rating), 2) AS average_rating
   FROM courses c
   LEFT JOIN feedback f ON f.course_id = c.id
   GROUP BY c.id"
);
foreach ($stmt as $row) {
  echo htmlspecialchars($row["code"]);
  echo " average: " . $row["average_rating"];
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Ders Geri Bildirim Formu: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$rating = (int)$_POST["rating"];
if ($rating >= 1 && $rating <= 5) {
  $stmt = $pdo->prepare(
    "INSERT INTO feedback (course_id, rating, comment)
     VALUES (?, ?, ?)"
  );
  $stmt->execute([
    (int)$_POST["course_id"], $rating, trim($_POST["comment"])
  ]);
}
```

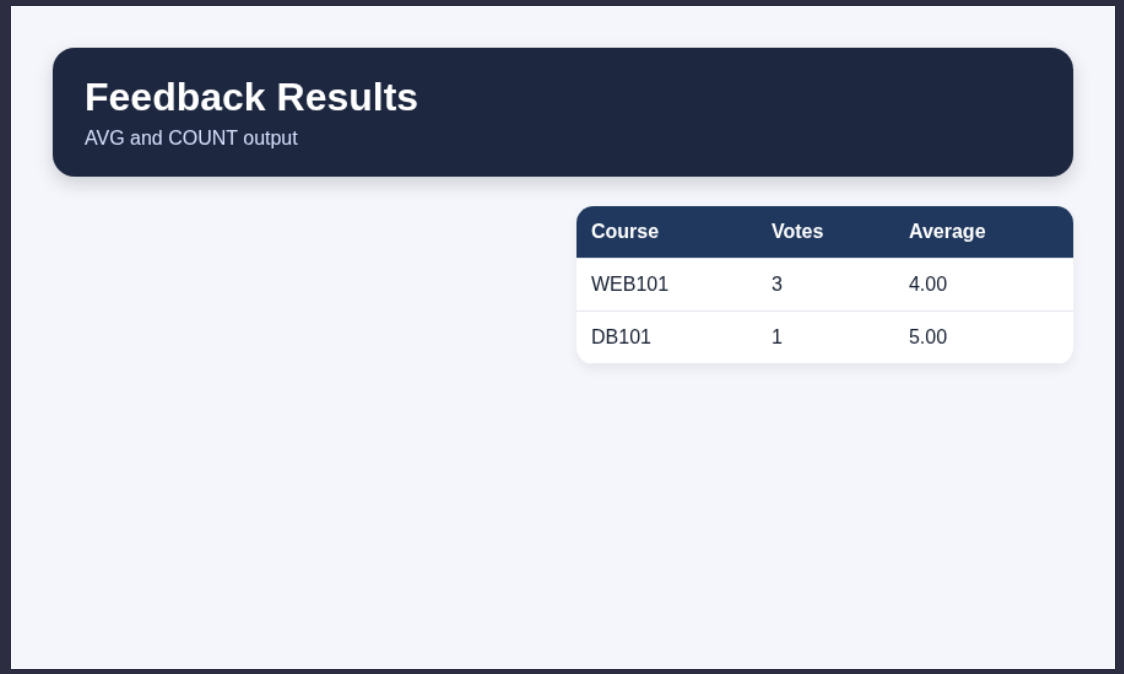
DB ÖNCESİ / SONRASI

```
BEFORE feedback
id | course_id | rating | comment
1  | 1         | 5      | Very useful
2  | 1         | 4      | More practice please

POST: course_id=1, rating=3, comment=Need more examples

AFTER feedback
id | course_id | rating | comment
3  | 1         | 3      | Need more examples
```

Ders Geri Bildirim Formu: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Aynı tarayıcıdan tekrar oy engeli için cookie ekle
- Yorumları tarihe göre sırala
- En düşük puanlı dersleri raporla

Mini Site 9: Mini Blog ve Yorum Paneli

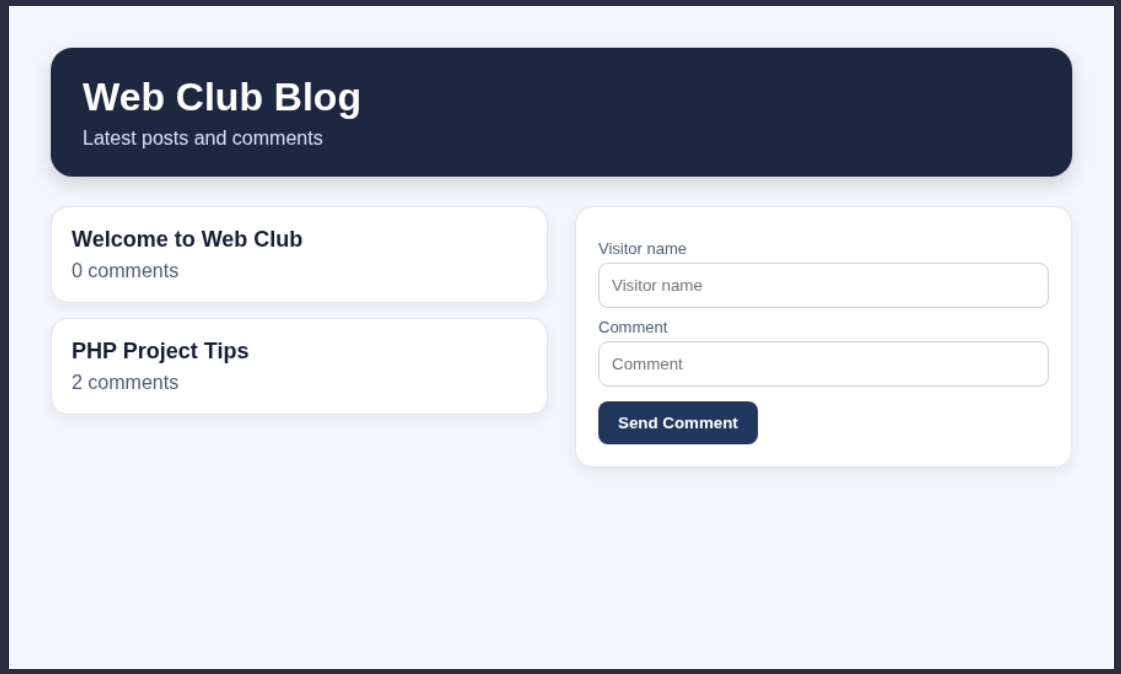
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Bir kulüp haberlerini yayınlar. Yönetici yazı ekler; ziyaretçiler yazıları okur ve yorum bırakır. Ana sayfada son yazılar gösterilir.

Öğrenci bu mini sitede neyi birleştirir?

- Slug/detail page mantığı
- INSERT ile yorum kaydı
- JOIN ile yorum sayısı
- LIMIT ile son içerikler



Beklenen çıktı: ana ekran / kullanıcı formu

Mini Blog ve Yorum Paneli: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE posts (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(120), slug VARCHAR(140), body TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE comments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  post_id INT, visitor_name VARCHAR(80), comment TEXT,
  FOREIGN KEY (post_id) REFERENCES posts(id)
);
```

DB ÖRNEK VERİ

```
posts
id | title                | slug
1  | Welcome to Web Club  | welcome-web-club
2  | PHP Project Tips     | php-project-tips

comments
id | post_id | visitor_name | comment
1  | 2       | Duru         | Great tips!
```

Mini Blog ve Yorum Paneli: Dosya akışı

Dosyalar ve görevleri

- index.php: son yazılar
- post.php: detay + yorum formu
- save-comment.php: yorum INSERT
- admin-new-post.php

REQUEST AKIŞI

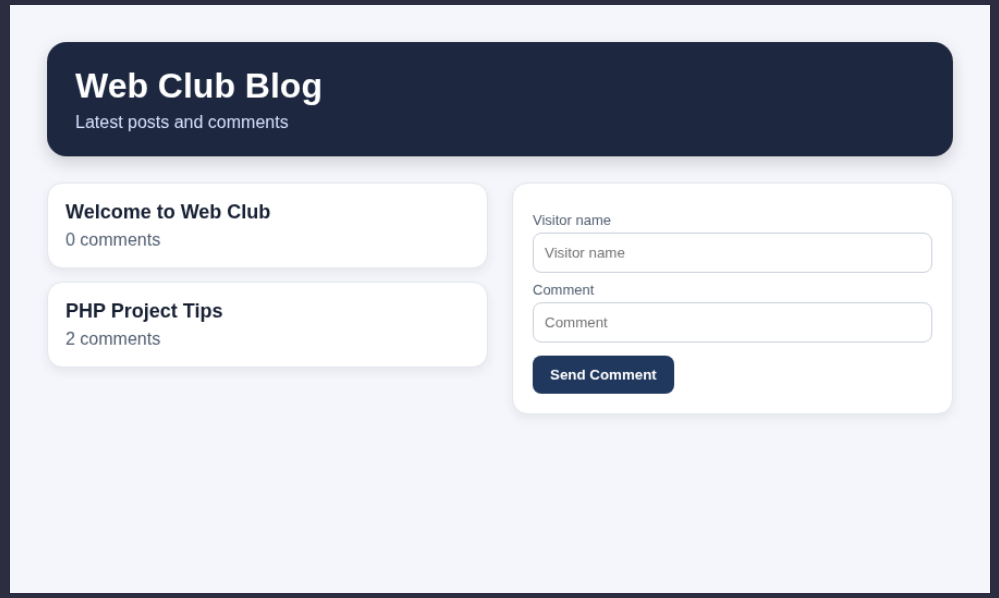
```
GET /index.php -> posts + comment_count
GET /post.php?slug=php-project-tips -> tek yazı
POST /save-comment.php -> comments INSERT
GET /admin-new-post.php -> yazı formu ve POST
kayıt
```

Mini Blog ve Yorum Paneli: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<?php foreach ($posts as $post): ?>
  <article>
    <h2>
      <a href="post.php?slug=<?php echo
        urlencode($post["slug"]); ?>">
        <?php echo htmlspecialchars($post["title"]); ?>
      </a>
    </h2>
    <p><?php echo htmlspecialchars(substr($post["body"], 0,
      90)); ?>...</p>
  </article>
<?php endforeach; ?>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Mini Blog ve Yorum Paneli: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
article { border-bottom: 1px solid #ccc; padding:
12px
  0; }
a { text-decoration: none; }
.comment { margin-left: 16px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

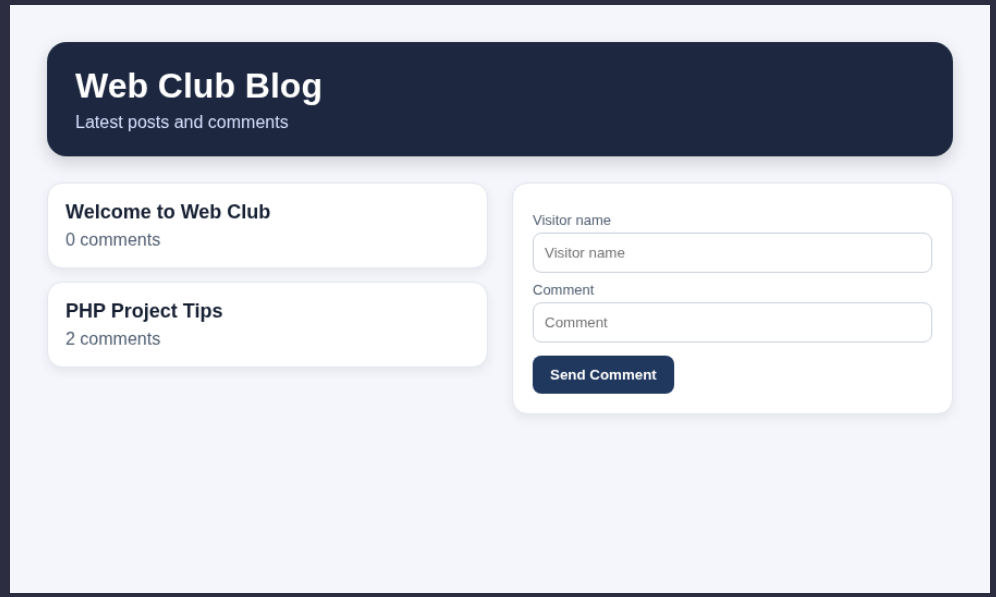
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Mini Blog ve Yorum Paneli: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
const textarea =
  document.querySelector("textarea[name='comment']");
const counter = document.querySelector("#counter");
textarea.addEventListener("input", function() {
  counter.textContent = this.value.length + " characters";
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Mini Blog ve Yorum Paneli: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$stmt = $pdo->query(
  "SELECT p.id, p.title, p.slug, COUNT(c.id) AS comment_count
   FROM posts p
   LEFT JOIN comments c ON c.post_id = p.id
   GROUP BY p.id
   ORDER BY p.created_at DESC
   LIMIT 5"
);
foreach ($stmt as $post) {
  echo htmlspecialchars($post["title"]);
  echo " (" . $post["comment_count"] . " comments)";
}
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Mini Blog ve Yorum Paneli: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO comments (post_id, visitor_name, comment)
   VALUES (?, ?, ?)"
);
$stmt->execute([
  (int)$_POST["post_id"],
  trim($_POST["visitor_name"]),
  trim($_POST["comment"])
]);

$post = $pdo->prepare("SELECT * FROM posts WHERE slug=?");
$post->execute([$_GET["slug"]]);
```

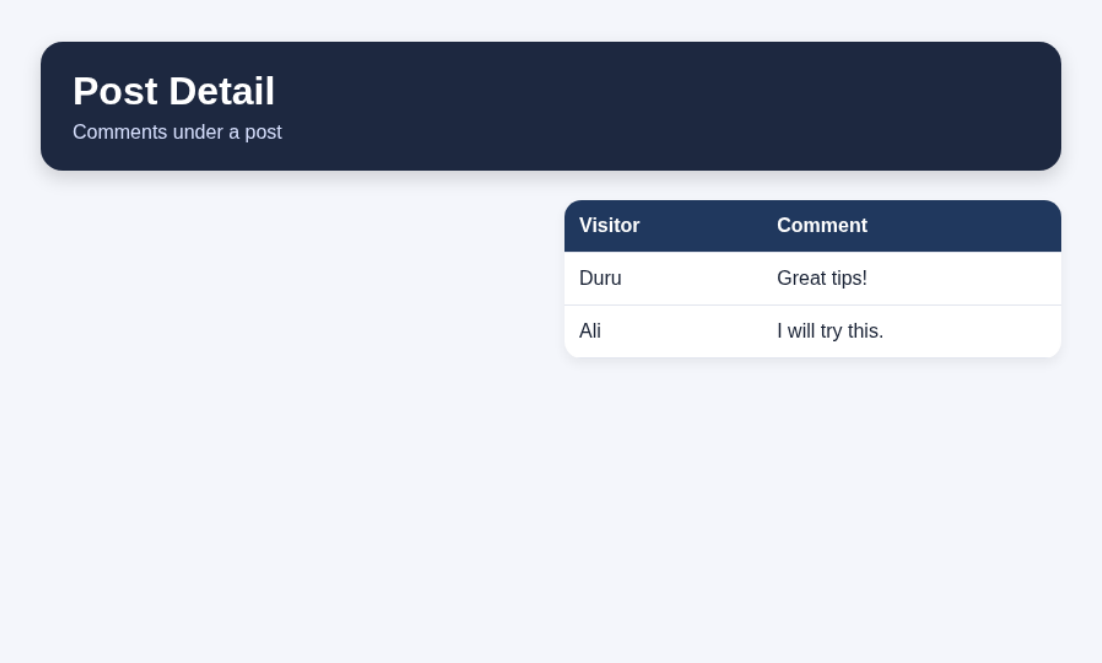
DB ÖNCESİ / SONRASI

```
BEFORE comments
id | post_id | visitor | comment
1  | 2       | Duru    | Great tips!

POST comment: post_id=2, visitor=Ali

AFTER comments
id | post_id | visitor | comment
2  | 2       | Ali     | I will try this.
```

Mini Blog ve Yorum Paneli: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- Slug benzersizliği kontrolü ekle
- Yorum onay sistemi ekle
- Admin yazı silme butonu ekle

Mini Site 10: Kayıp-Eşya İlan Panosu

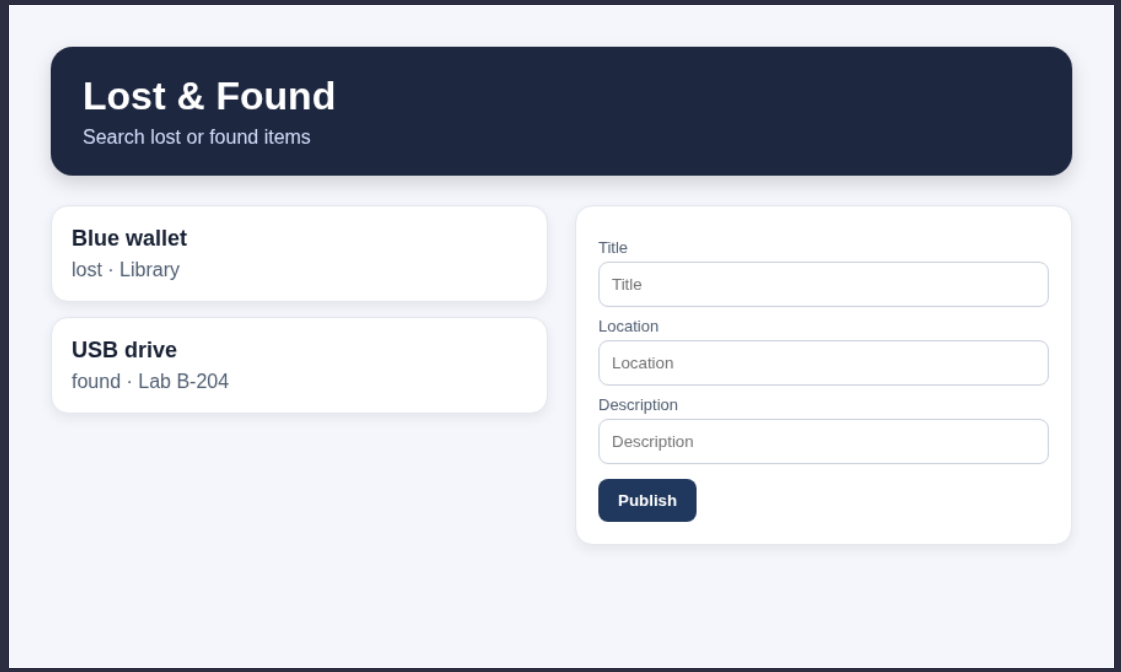
Hikaye, amaç ve beklenen ilk ekran

Hikaye

Kampüste kayıp ve bulunan eşyalar için basit bir ilan panosu hazırlanır. Kullanıcı ilan ekler, arama yapar; görevli ilanı çözüldü olarak işaretler.

Öğrenci bu mini sitede neyi birleştirir?

- GET ile arama ve tür filtresi
- POST ile ilan ekleme
- UPDATE ile solved durumu
- Dosya/DB ayrımı fikri



Beklenen çıktı: ana ekran / kullanıcı formu

Kayıp-Eşya İlan Panosu: Veritabanında ne var?

Şema + örnek satırlar

SQL ŞEMA

```
CREATE TABLE items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  item_type VARCHAR(10), title VARCHAR(100), location
    VARCHAR(80),
  description TEXT, contact_email VARCHAR(120), solved
    TINYINT DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

DB ÖRNEK VERİ

```
items
id | item_type | title        | location  | solved
1  | lost      | Blue wallet  | Library   | 0
2  | found     | USB drive    | Lab B-204 | 0
3  | found     | Water bottle | Cafeteria | 1
```

Kayıp-Eşya İlan Panosu: Dosya akışı

Dosyalar ve görevleri

- items.php: ilan listesi ve arama
- new-item.php: ilan formu
- save-item.php: INSERT
- solve-item.php: UPDATE

REQUEST AKIŞI

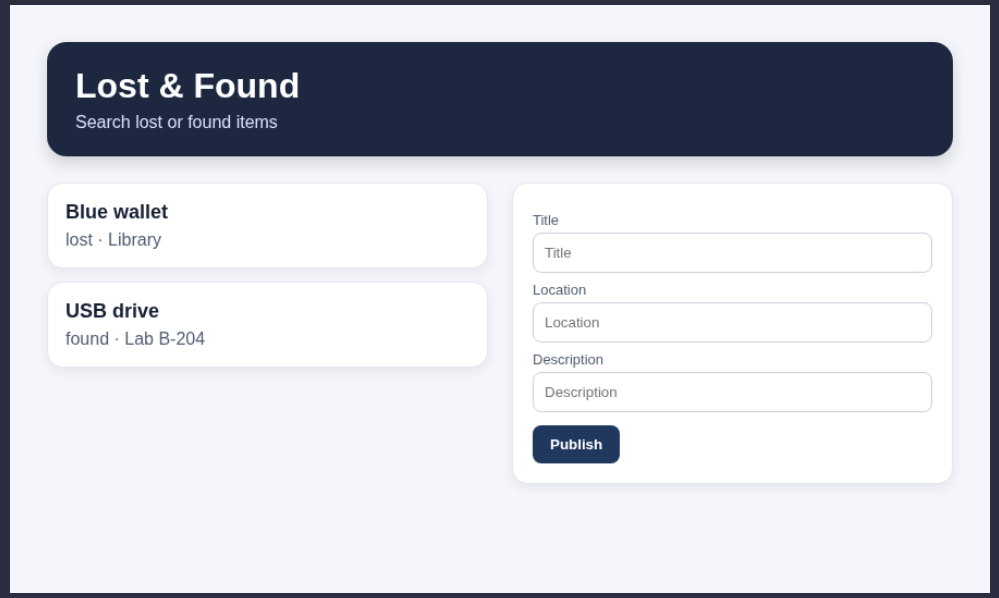
```
GET /items.php?q=wallet -> arama
GET /items.php?item_type=found -> tür filtresi
POST /save-item.php -> items INSERT
POST /solve-item.php -> solved UPDATE
```

Kayıp-Eşya İlan Panosu: HTML içinde PHP çıktısı

Form, liste ve echo kullanımı

index.php ÖRNEĞİ

```
<h1>Lost & Found</h1>
<form method="get" action="items.php">
  <input name="q" placeholder="Search title or location">
  <select name="item_type">
    <option
      value="">All</option><option>lost</option><option>found</option>
  </select>
  <button>Search</button>
</form>
<form method="post" action="save-item.php">
  <input name="title" required>
  <input name="location" required>
  <textarea name="description"></textarea>
  <button>Publish</button>
</form>
```



Dikkat

Veri ekrana basılırken echo + htmlspecialchars kullanılır.

Kayıp-Eşya İlan Panosu: CSS sadece okunabilirlik için

assets/style.css

```
body { font-family: Arial; margin: 24px; }
.item { border: 1px solid #ccc; margin: 8px 0;
padding:
  10px; }
.solved { opacity: 0.6; }
input, select, textarea, button { margin: 5px;
padding:
  7px; }
```

Beklenti

Bu içerikte CSS ana konu değildir; amaç HTML çıktısını okunabilir hale getirmektir.

Öğrenciden istenen

Form alanları alt alta gelmeli, tablo okunmalı, hata/başarı mesajı basitçe ayırt edilmelidir.

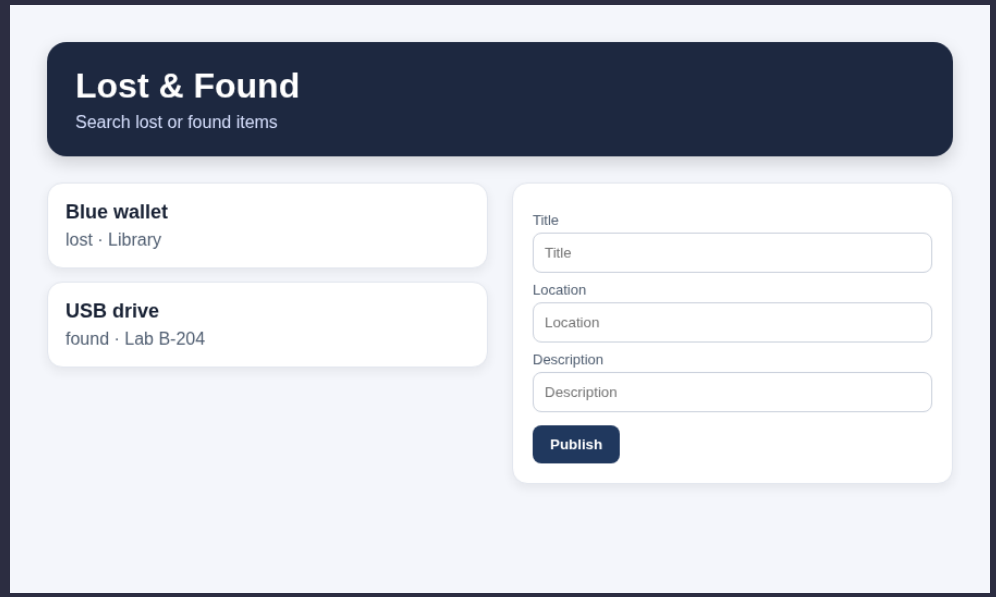
Yapılmayacak şey

Hazır UI kütüphanesine veya karmaşık animasyonlara gerek yok.

Kayıp-Eşya İlan Panosu: JavaScript küçük ama anlamlı olmalı

assets/app.js

```
document.querySelector("textarea[name='description']").addEventLis
tener("input",
  function() {
  const left = 160 - this.value.length;
  document.querySelector("#left").textContent = left + "
    characters left";
});
```



Kural

JS kullanıcı deneyimini iyileştirir; asıl kayıt ve güvenlik kontrolü PHP tarafında tekrar yapılır.

Kayıp-Eşya İlan Panosu: SELECT / JOIN / GROUP BY çıktısı

PHP + SQL OKUMA

```
$q = $_GET["q"] ?? "";
$type = $_GET["item_type"] ?? "";
$sql = "SELECT * FROM items
        WHERE solved = 0
        AND (title LIKE ? OR location LIKE ?)
        AND (? = '' OR item_type = ?)";
$stmt = $pdo->prepare($sql);
$stmt->execute(["%$q%", "%$q%", $type, $type]);
```

Sorgu sonucunu okuma

Sorgu sonucunda öğrencinin görmesi gereken:

- DB'deki ham satırlar tek tek seçilir.
- JOIN varsa satırlar başka tabloyla eşleşir.
- GROUP BY varsa satırlar özetlenir.
- echo ile güvenli HTML çıktısı oluşur.

Öğrenci çıktıyı DB satırlarıyla eşleştirebilmelidir.

Kayıp-Eşya İlan Panosu: POST / UPDATE / DELETE etkisi

Öncesi-sonrası ile veritabanı değişimini görme

PHP İŞLEM DOSYASI

```
$stmt = $pdo->prepare(
  "INSERT INTO items
   (item_type, title, location, description, contact_email)
   VALUES (?, ?, ?, ?, ?)"
);
$stmt->execute([
  $_POST["item_type"], trim($_POST["title"]),
    trim($_POST["location"]),
  trim($_POST["description"]), trim($_POST["contact_email"])
]);

$solve = $pdo->prepare("UPDATE items SET solved=1 WHERE id=?");
$solve->execute([(int)$_POST["id"]]);
```

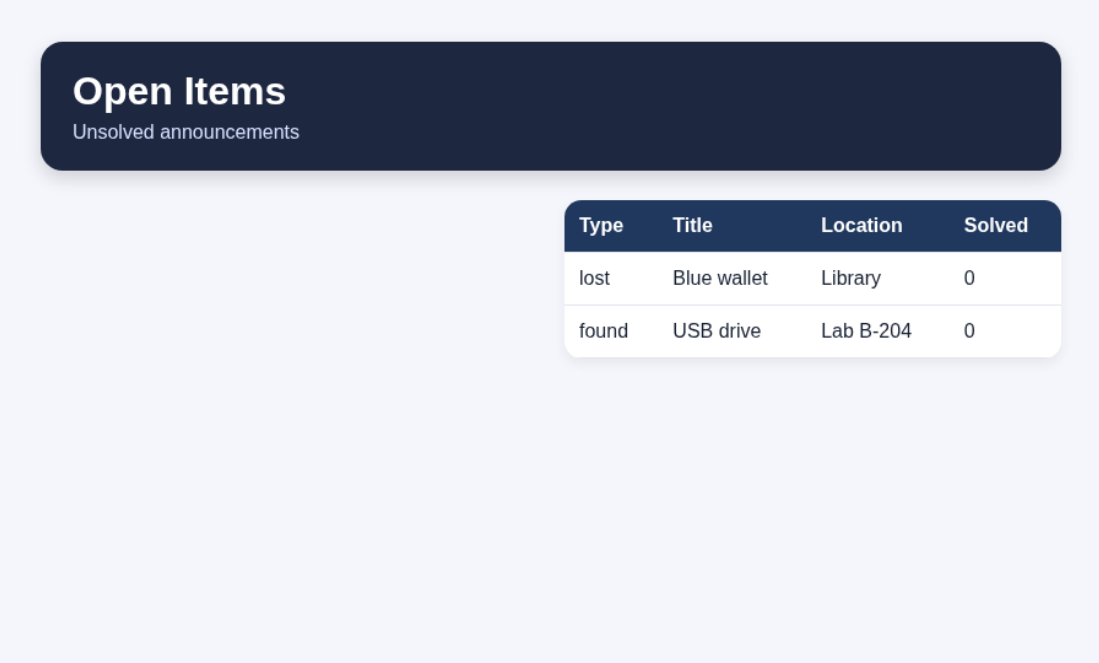
DB ÖNCESİ / SONRASI

```
BEFORE items
id | title       | solved
1  | Blue wallet | 0
2  | USB drive   | 0

POST solve: id=1

AFTER items
id | title       | solved
1  | Blue wallet | 1
2  | USB drive   | 0
```

Kayıp-Eşya İlan Panosu: Entegrasyon testi ve geliştirme fikirleri



Beklenen çıktı: admin/liste/rapor ekranı

Kontrol listesi

Test akışı:
1. index.php ekranında DB verisi görünüyor mu?
2. Form gönderilince doğru tablo değişiyor mu?
3. UPDATE/DELETE sonrası liste yenileniyor mu?
4. Hatalı input durumunda mesaj çıkıyor mu?
5. echo ile basılan değerler güvenli mi?

Ek geliştirme fikirleri

- İlan fotoğrafı için dosya yükleme ekle
- E-posta doğrulama yap
- Çözülen ilanları ayrı arşiv sayfasında göster