Building a Security Dashboard Using PHP and MySQL

This standalone example queries fictional security-event data with PDO and presents useful summary and detail output.

A dashboard is meaningful when each output answers a user question. Do not add a statistic or chart merely because it is easy to calculate.

Connect with PDO

Use the shared connection created in `includes/database.php`:

```
<?php
declare(strict_types=1);


require __DIR__ . '/includes/database.php';
```

Do not repeat database credentials in the dashboard page.

Query useful summaries

```
$totalEvents = (int) $pdo
    ->query('SELECT COUNT(*) FROM security_events')
    ->fetchColumn();

$highRiskEvents = (int) $pdo
    ->query(
        "SELECT COUNT(*)
         FROM security_events
```

```
         WHERE severity IN ('high', 'critical')"
    )
    ->fetchColumn();

$deniedEvents = (int) $pdo
    ->query(
        "SELECT COUNT(*)
         FROM security_events
         WHERE access_result IN ('denied', 'blocked')"
    )
    ->fetchColumn();
```

These queries answer three different questions: how much activity exists, how much is high priority, and how often access was prevented.

Retrieve recent records

```
$recentStatement = $pdo->query(
    'SELECT
        event_timestamp,
        device_id,
        room,
        event_type,
        severity,
        access_result
     FROM security_events
     ORDER BY event_timestamp DESC
     LIMIT 10'
);

$recentEvents = $recentStatement->fetchAll();
```

Select only the fields needed by the interface.

Present accessible summary output

```
<section aria-labelledby="summary-heading">
  <h1 id="summary-heading">Security event summary</h1>

  <div class="summary-grid">
    <article class="card">
      <h2>Total events</h2>
      <p><?= $totalEvents ?></p>
    </article>

    <article class="card">
      <h2>High-priority events</h2>
      <p><?= $highRiskEvents ?></p>
    </article>

    <article class="card">
      <h2>Denied or blocked</h2>
      <p><?= $deniedEvents ?></p>
    </article>
  </div>
</section>
```

Headings and text communicate the meaning without relying on colour alone.

Present recent events safely

```
<?php if (!$recentEvents): ?>
  <p>No fictional events are available.</p>
<?php else: ?>
  <div class="table-wrap">
    <table>
      <caption>Ten most recent fictional security events</caption>
      <thead>
        <tr>
          <th scope="col">Time</th>
          <th scope="col">Device</th>
          <th scope="col">Room</th>
          <th scope="col">Event</th>
          <th scope="col">Severity</th>
```

```
          <th scope="col">Access result</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($recentEvents as $event): ?>
          <tr>
            <td><?= htmlspecialchars($event['event_timestamp']) ?></td>
            <td><?= htmlspecialchars($event['device_id']) ?></td>
            <td><?= htmlspecialchars($event['room']) ?></td>
            <td><?= htmlspecialchars($event['event_type']) ?></td>
            <td><?= htmlspecialchars($event['severity']) ?></td>
            <td><?= htmlspecialchars($event['access_result']) ?></td>
          </tr>
        <?php endforeach; ?>
      </tbody>
    </table>
  </div>
<?php endif; ?>
```

`htmlspecialchars()` prevents stored text from being interpreted as page markup.

Responsive styling

```
.summary-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(12rem, 1fr));
    gap: 1rem;
}

.table-wrap {
    overflow-x: auto;
}

table {
    width: 100%;
    border-collapse: collapse;
}
```

```
th,
td {
    padding: 0.75rem;
    border: 1px solid #cbd5e1;
    text-align: left;
}
```

Test the dashboard

Confirm that:

- each total matches an independent SQL check;
- an empty database has a clear message;
- text is escaped;
- the table remains usable on a narrow screen;
- headings explain each value; and
- all records are fictional.

Revision #3
Created 2026-06-10 01:58:21 UTC by Mr Napper
Updated 2026-08-19 23:03:35 UTC by Mr Napper