Attchme

🔐 1. Advanced Measures For Attachme

Protect the database from **hacking, SQL injection, unauthorized access, and data leaks**.

1️⃣ Restrict Root Access & Create a Secure Database User

By default, MySQL in XAMPP uses **root** without a password. **Change this** for security.

✅ Steps to Create a New Secure User in phpMyAdmin

1. Open **phpMyAdmin** (`http://localhost/phpmyadmin`).
2. Click **User Accounts** → **Add User Account**.
3. Enter:
   - **Username:** `attachme_admin`
   - **Host:** `localhost`
   - **Password:** `StrongP@ssw0rd!` (Use a strong password).
4. Grant **Only Required Privileges** (Avoid full admin rights).
5. Click **Go** to create the user.

✅ Steps to Set a Password for Root User

1. In phpMyAdmin, go to **User Accounts**.
2. Find `root@localhost` → Click **Edit Privileges**.
3. Under **Change Password**, enter a strong password.
4. Click **Go** and restart MySQL in XAMPP.

🚀 **Result**:

- **Prevents unauthorized access** to MySQL.
- **Stops attackers from using default credentials.**

2️⃣ Prevent SQL Injection with Prepared Statements

Instead of raw queries like this:

php
CopyEdit
```
$sql = "SELECT * FROM users WHERE email = '$email'";
```

✅ **Use Secure Prepared Statements in PHP:**

php
```
$conn = new PDO("mysql:host=localhost;dbname=attachme_db",
"attachme_admin", "StrongP@ssw0rd!");
$stmt = $conn->prepare("SELECT * FROM users WHERE email = :email");
$stmt->execute([":email" => $email]);
$user = $stmt->fetch();
```

🚀 **Result**:

- Prevents **SQL injection attacks**.
- Ensures **data integrity**.

3️⃣ Enable SSL Encryption for Secure Database Communication

✅ **Steps to Enable SSL in MySQL (XAMPP on Linux)**

Open MySQL config file:
bash

```
sudo nano /opt/lampp/etc/my.cnf
```

1.

Add this under `[mysqld]`:
ini
```
require_secure_transport=ON
```

2. Restart MySQL:
   bash
   CopyEdit
   ```
   sudo /opt/lampp/lampp restartmysql
   ```
3. 🚀 **Result**:

- **Encrypts data traffic** between PHP and MySQL.
- Prevents **man-in-the-middle (MITM) attacks**.

4️⃣ Restrict Direct Database Access

By default, MySQL allows **remote connections**, which is **risky**.

✅ **Steps to Disable Remote Access:**

1. Open **phpMyAdmin** → **User Accounts**.
2. Find **root@%** (any host) → **Delete It** (Only keep `root@localhost`).
3. Restart MySQL.

🚀 **Result**:

- Only **local applications** (PHP) can access MySQL.
- Blocks **external hacking attempts**.

⚡ 2. Performance Optimization (For Fast Queries & Large Data Sets)

5️⃣ Indexing for Faster Queries

Indexing **speeds up** search queries.

✅ **Add Index to Frequently Searched Columns:**

sql
```
CREATE INDEX idx_email ON users(email);
CREATE INDEX idx_company_name ON companies(company_name);
CREATE INDEX idx_application_status ON applications(status);
```

🚀 **Result**:

- **Speeds up SELECT queries**.
- **Reduces query load on the database**.

6️⃣ Optimize Table Storage Engine (Use InnoDB)

By default, MySQL may use **MyISAM**, which is slower.

✅ **Convert Tables to InnoDB for Better Performance:**

sql

CopyEdit

```
ALTER TABLE users ENGINE = InnoDB;
ALTER TABLE applications ENGINE = InnoDB;
```

🚀 **Result**:

- **Faster transactions & better reliability.**
- Supports **foreign keys**.

7️⃣ Optimize Large Data Queries

✅ Use Pagination in PHP to Prevent Overloading

php

```
$limit = 10;
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
$start = ($page - 1) * $limit;

$stmt = $conn->prepare("SELECT * FROM applications LIMIT :start,
:limit");
$stmt->bindValue(":start", $start, PDO::PARAM_INT);
$stmt->bindValue(":limit", $limit, PDO::PARAM_INT);
$stmt->execute();
$results = $stmt->fetchAll();
```

🚀 **Result**:

- Prevents **loading thousands of rows at once**.
- Faster **page load times**.

8️⃣ Optimize Foreign Key Relationships

✅ Use `ON DELETE CASCADE` to Auto-Remove Related Data

sql

```
ALTER TABLE students ADD CONSTRAINT fk_student_user FOREIGN KEY
(user_id) REFERENCES users(id) ON DELETE CASCADE;
```

🚀 **Result**:

- **Auto-cleans up orphaned data** (e.g., student applications get deleted if the student is removed).

📈 3. Scalability (Handling Growth & Large User Bases)

🔟 Connection Pooling (To Handle High Traffic)

Instead of opening new connections for every query, use **persistent connections**.

✅ In PHP (PDO)

php
```
$pdo = new PDO("mysql:host=localhost;dbname=attachme_db",
"attachme_admin", "StrongP@ssw0rd!", [
    PDO::ATTR_PERSISTENT => true
]);
```


🚀 **Result**:

- **Reduces server load** during high traffic.

1️⃣1️⃣ Enable Query Caching

✅ Enable MySQL Query Cache for Repeated Queries

Open `my.cnf`:
bash
```
sudo nano /opt/lampp/etc/my.cnf
```

1.

Add:
ini
```
[mysqld]
query_cache_type=1
query_cache_size=64M
```

2.

Restart MySQL:
bash
```
sudo /opt/lampp/lampp restartmysql
```

3.

🚀 **Result**:

- **Speeds up repeated queries** by caching results.

🔹 Summary of Key Implementations

Category	Advanced Implementation in MySQL & PHP
🔒 **Security**	Use hashed passwords, SSL, restrict root access, prepared statements
⚡ **Performance**	Use indexes, query optimization, pagination, InnoDB
🔄 **Scalability**	Partitioning, connection pooling, query caching