جامـــــعة المـــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قســـم الانظمة الطبية الذكية

Lecture: 5

Subject: **Working with Databases, Form Handling, Data Validation, and Adding New Records**

Level: Third stage
Lecturer: Msc najwan thaeer ali

Building a robust user management system is the backbone of any secure application. It's the gatekeeper that ensures only the right people get in and that their data stays safe once they do. From the moment a user signs up to the daily management of their profile, every step requires a mix of **smooth user experience** and **hardcore security**. At its core, this process is about trust:

- **Identity:** Proving the user is who they say they are.
- **Integrity:** Ensuring the data they provide is accurate and untampered.
- **Security:** Keeping that data locked down against unauthorized access.

Would you like to dive into the technical details for **front-end validation** or **back-end security**?

🎨 Front-End Validation

🔒 Back-End Security

🎯 General Objectives

1. Understand how web applications interact with databases.
2. Learn how to handle HTML form submissions securely.
3. Apply proper data validation techniques.
4. Perform database operations to add new records safely.

🎯 Behavioral Objectives

By the end of this lecture, students should be able to:

- Explain the concept of database connectivity.
- Design a basic HTML form.
- Differentiate between client-side and server-side validation.
- Write SQL statements to insert new records.
- Apply security measures when inserting data.

1️⃣ Introduction to Databases in Web Applications

A **database** is an organized collection of data stored electronically.

In web systems, databases are used to store:

- User accounts
- Products
- Orders
- Academic records
- Messages

The most common type used in web systems is the **Relational Database**, such as:

- MySQL
- PostgreSQL
- SQL Server
- Oracle

2️⃣ Database Connection Process

Before performing any operation, the application must:

1. Connect to the database server.
2. Select the database.
3. Execute SQL queries.
4. Close the connection.

Example Flow:

User submits form → Server processes data → Insert into database → Return response

---

3️⃣ Form Handling (Processing User Input)

Forms are the primary method of collecting user input.

Example: Student Registration Form

Fields may include:

- Full Name
- Email
- Phone Number
- Password

Steps in Form Handling:

1. User fills the form.
2. Clicks Submit.
3. Data is sent to the server.
4. Server validates the data.
5. If valid → Save to database.
6. If invalid → Return error message.

▨ 4️⃣ Data Validation

Validation ensures that data is correct before storing it.

◇ Types of Validation

1. Client-Side Validation

- Done in the browser.
- Improves user experience.
- Example: Required fields, email format.

2. Server-Side Validation (Most Important)

- Done on the server.
- Prevents malicious input.
- Must always be implemented.

◇ Examples of Validation Rules

Field	Validation Rule
Name	Cannot be empty
Email	Must contain @
Password	Minimum 8 characters
Age	Must be numeric

▨ 5️⃣ Database Operations – Adding New Records

The main SQL command used to add new records is:

◇ INSERT Statement

```
INSERT INTO Students (Name, Email, Phone)
VALUES ('Ali Ahmed', 'ali@mail.com', '0770000000');
```

◇ General Syntax

INSERT INTO table_name (column1, column2, column3)
VALUES (value1, value2, value3);

◇ Example: User Registration

INSERT INTO Users (FullName, Email, Password)
VALUES ('Sara Ali', 'sara@mail.com', 'hashed_password');
⚠ Important: Password must be hashed before storing.

▨ 6 Security Considerations When Adding Records

🔐 1. Prevent SQL Injection

Bad Example:
SELECT * FROM Users WHERE Email = '$email';
Secure Method:

- Use Prepared Statements
- Use Parameterized Queries

🔐 2. Hash Passwords

Never store passwords in plain text.
Use hashing algorithms such as:

- bcrypt
- SHA-256

🔐 3. Sanitize Input

Remove:

- Script tags
- Special SQL characters
- Malicious code

▨ 7 Complete Workflow Example

1. User opens registration page.

2. Fills the form.
3. Client-side validation checks required fields.
4. Server receives data.
5. Server performs validation again.
6. Password is hashed.
7. Data inserted into database using prepared statement.
8. Success message displayed.

▨ 8️⃣ Common Errors When Adding Records

- Forgetting to validate input.
- Allowing duplicate email addresses.
- Not hashing passwords.
- Not handling database connection errors.

▧ 9️⃣ Best Practices

- Always validate on server side.
- Use HTTPS.
- Use unique constraints in database.
- Use transactions when needed.
- Log database errors securely.

These core components of a user management system ensure that identities are verified, data is handled securely, and user interactions remain seamless.

1. Account Activation Processes

Account activation confirms a user's intent and identity before granting full access.

- **Email Confirmation:** Sending a verification link or a unique token to the user’s email address is the standard method for confirming ownership.

- **Multi-Channel Tokens:** Using NetIQ, systems can send activation tokens via SMS, email, or both for added security.

- **Dormant Account Re-activation:** In some sectors, like banking, re-activating a dormant account may require submitting updated KYC (Know Your Customer) documents for manual verification.

2. User Profile Management

This involves managing a structured representation of a user's characteristics and preferences.

- **Core Attributes:** Profiles typically store essential data like name, profile pictures, and contact details.
- **Role-Based Access Control (RBAC):** Users are assigned specific roles (e.g., admin, moderator, regular user) that determine their system privileges.
- **Self-Service Options:** Users should have the ability to update their personal information, change passwords, and manage privacy settings through a dedicated "My Profile" section.

3. Handling Form Submissions

Effective form handling ensures data is correctly captured and securely transmitted to the backend.

- **Submission Methods:** Use the `POST` method for sensitive data like passwords to ensure they are sent in the HTTP request body rather than appended to the URL.
- **Preventing Default Behavior:** In modern frameworks like React, developers use `preventDefault()` to handle data via JavaScript without a full page reload.
- **Data Processing:** The backend validates inputs, prevents spam (using CAPTCHAs or honeypots), and triggers automated responses like confirmation emails.

4. Database Operations & Data Integrity

Performing operations on user records requires strict adherence to security and integrity protocols.

- **CRUD Operations:** Systems must enforce specific authorizations—such as Read, Insert, and Delete—to ensure users can only perform permitted actions on data objects.
- **Integrity Constraints:** Use database triggers and constraints to enforce business rules, such as preventing duplicate records or ensuring data falls within valid ranges.
- **Regular Maintenance:** Implement scheduled backups, application updates, and regular security audits to protect against data loss and zero-day exploits.

5. Validation & Security Measures

Security measures protect sensitive user data from unauthorized access and tampering.

- **Authentication:** Multi-factor authentication (MFA) is a critical standard, often making a system 99% less likely to be hacked.
- **Encryption:** Data should be encrypted both at rest (using protocols like TDE) and in transit (using TLS).
- **Least Privilege Principle:** Users should only be granted the minimum access level necessary to perform their specific job functions.