جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الانظمة الطبية الذكية

Lecture: 5

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

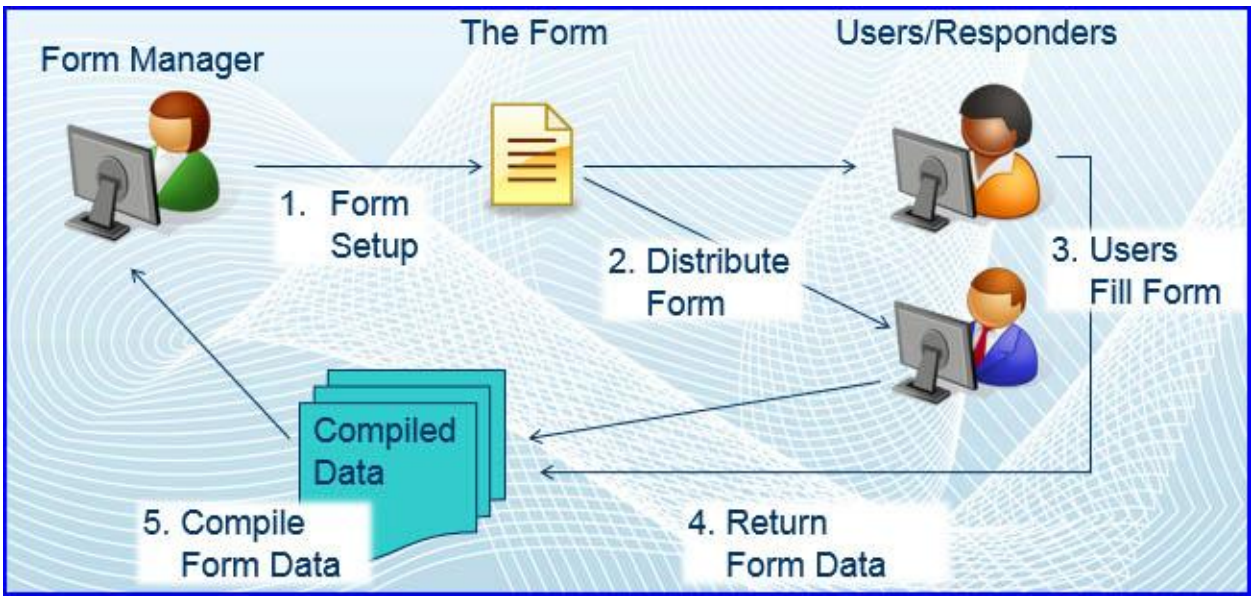
Level: Third stage

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



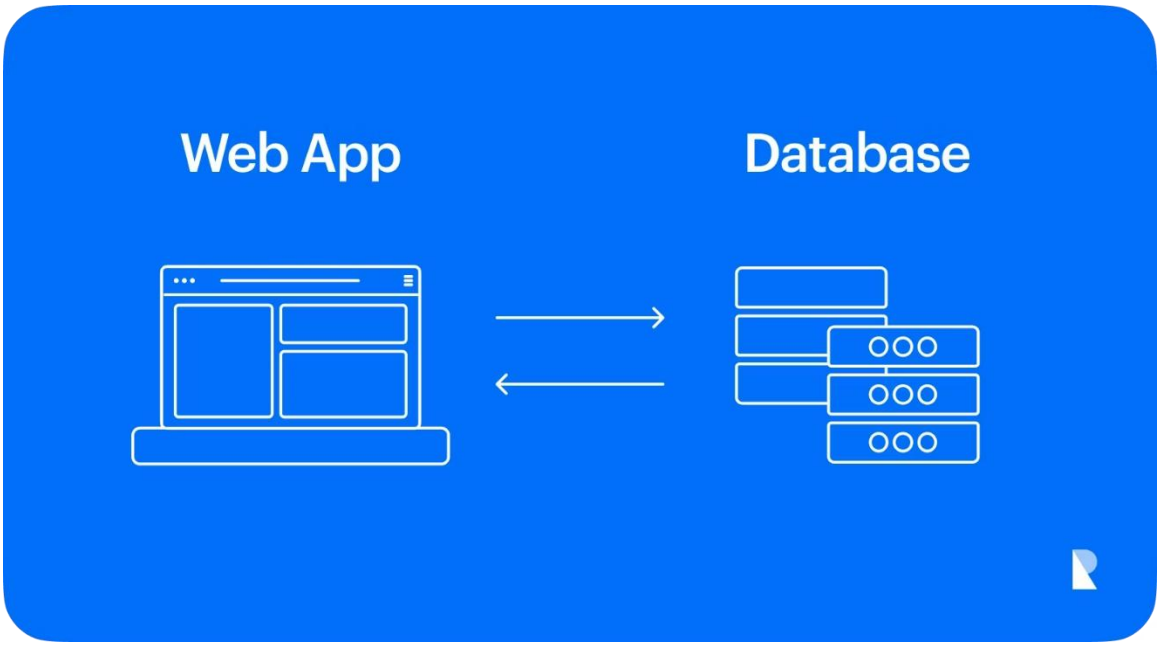
General Objectives

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.

Introduction to Databases in Web Applications A **database** is an organized collection of data stored electronically.

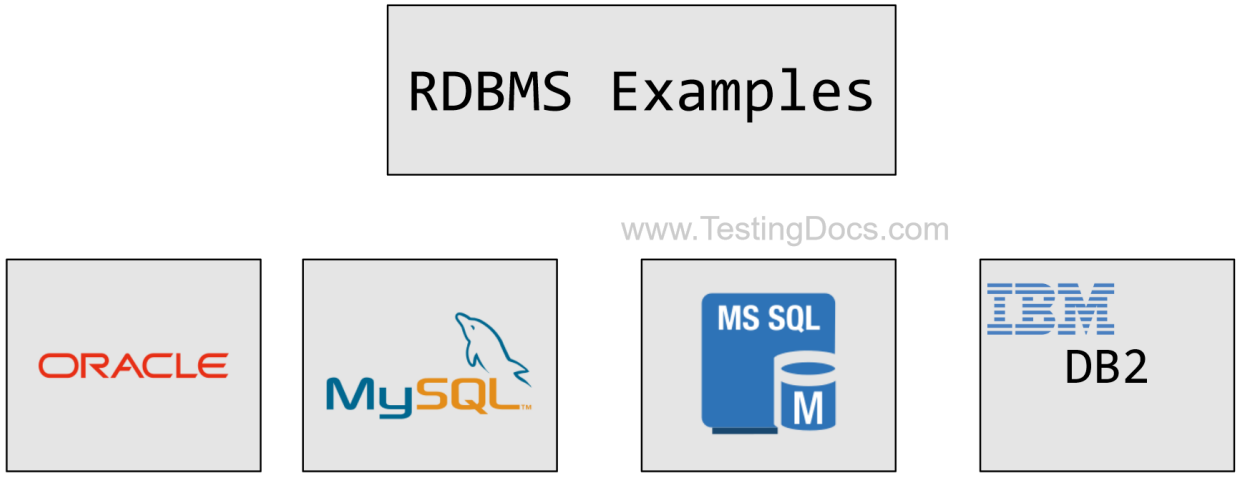
In web systems, databases are used to store:

- 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



Relational Database Management Systems

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

Form Handling (Processing User Input) Forms are the primary method of collecting user input

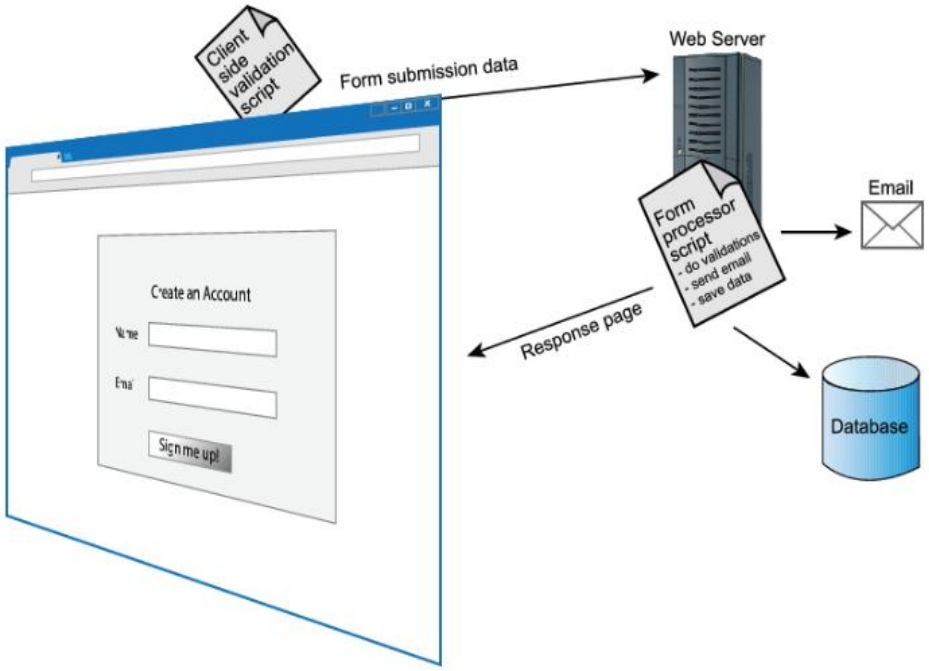
. **Example: Student Registration Form** Fields may include:

• 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.

Headings Are Important

Search engines use the headings to index the structure and content of your web pages.

Users often skim a page by its headings. It is important to use headings to show the document structure.

<h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

For example:

- <h1> - Page title
- <h2> - Section titles
- <h3> - Sub-sections

```
<!DOCTYPE html>
<html>
<body>

<h1>Travel Guide</h1>

<h2>Europe</h2>
<h3>France</h3>
<h3>Italy</h3>

<h2>Asia</h2>
<h3>India</h3>
<h3>Thailand</h3>

</body>
</html>
```

Travel Guide

Europe

France

Italy

Asia

India

Thailand

Data Validation ensures that data is correct before storing it.

- Types of Validation

1. Client-Side Validation

• Done in the browser.

• Improves user experience.

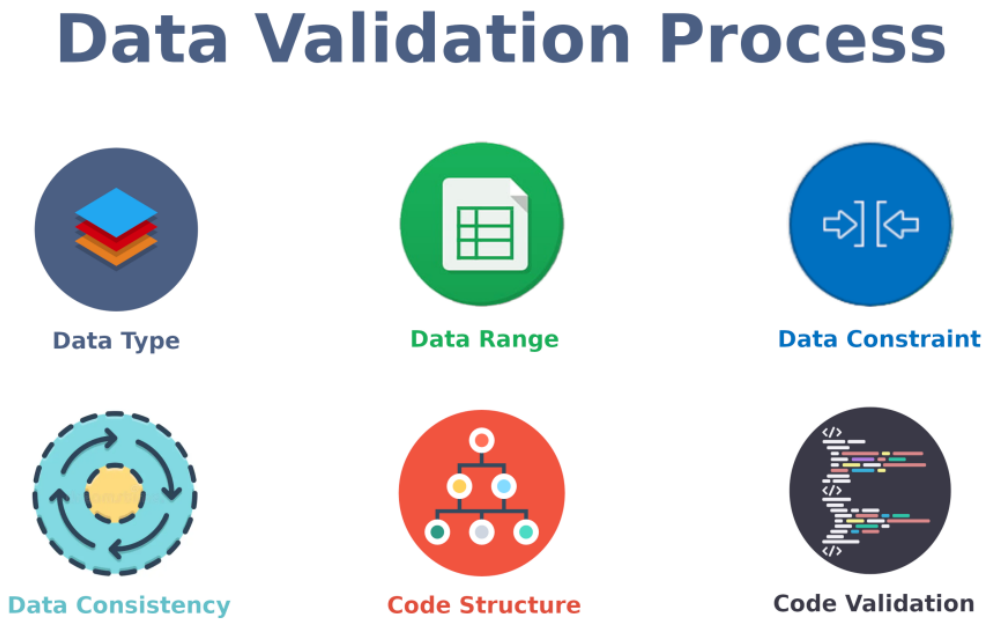
• Example: Required fields, email format.

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

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.

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