Date: 08-04-2025

Lab Manual for Experiment

Objective:

To implement form data storage in MongoDB using PHP and handle file uploads in both PHP and Node.js environments.Understand and use JavaScript data types, variables, and operators

Pre-requisite Tools:

- Local development environment (XAMPP/WAMP/MAMP or equivalent)
- MongoDB Community Server installed
- PHP MongoDB driver installed
- Node.js installed
- Text editor (VS Code, Sublime Text)
- Postman or similar API testing tool

Assignment :

1. **PHP Form Handling with MongoDB**
   Write a PHP script that creates an HTML form to collect user details (name, email, age) and inserts this data into a MongoDB collection named users. Ensure the form uses the POST method and includes basic validation (e.g., non-empty fields, valid email format).

2. **File Upload in PHP**
   Extend the PHP form from Question 1 to include a file upload field (e.g., profile picture). The script should:
   - Validate the file type (e.g., only allow JPEG/PNG).
   - Save the file to a directory on the server.
   - Store the file path in the MongoDB document along with other user details.

3. **Reading Form Data in Node.js**
   Create a Node.js script using Express.js to handle a POST request from an HTML form (name, email, message). The script should parse the form

data and log it to the console. Use middleware like `body-parser` or `express.urlencoded()`.

4. **File Upload in Node.js**
   Modify the Node.js script from Question 3 to handle file uploads (e.g., a resume PDF). Use the `multer` middleware to:
   - Save the uploaded file to a designated folder.
   - Return a JSON response confirming the upload and displaying the file metadata (name, size, type).

5. **MongoDB Integration in Node.js**
   Write a Node.js script that connects to MongoDB and inserts form data (name, email, phone) into a collection named `contacts`. Include error handling for connection issues and duplicate entries.

6. **Combining File Upload and MongoDB in Node.js**
   Create a Node.js application where users can submit a form with a profile image. The application should:
   - Save the image to the server.
   - Store the user's details and image path in MongoDB.
   - Return a success message with the saved data.

7. **Form Data Validation in PHP**
   Enhance the PHP form from Question 1 to include server-side validation for:
   - Ensuring the email is unique (query MongoDB before insertion).
   - Validating age is a positive number.
   - Displaying error messages if validation fails.

8. **Form Data Validation in Node.js**
   Implement validation in the Node.js script from Question 5 to check:
   - Email format using regex.
   - Phone number length and digits.
   - Return appropriate HTTP status codes (e.g., 400 for invalid data).

9. **Displaying MongoDB Data in PHP**
   Write a PHP script that queries the `users` collection in MongoDB and displays the results in an HTML table. Include options to filter users by age or email domain (e.g., "@gmail.com").

10. **Pagination in Node.js with MongoDB**
    Modify the Node.js script from Question 6 to support pagination. The script should:
    - Accept query parameters for page number and limit (e.g., `?page=1&limit=5`).
    - Return paginated results from the MongoDB collection.
    - Include metadata like total pages and current page in the response.

Key Focus Areas:

- **PHP**: Form handling, file uploads, MongoDB CRUD, validation.
- **Node.js**: Reading form data, file uploads with multer, MongoDB integration, validation, and pagination.
- **Security**: Input sanitization, file type validation, and error handling in both PHP and Node.js.

Learning Outcomes:

After completing this lab session, students should be able to:

1. **PHP & MongoDB**
   - Perform CRUD operations using PHP and MongoDB.
   - Handle form data with server-side validation.
2. **File Uploads**
   - Upload files in PHP/Node.js and store metadata in MongoDB.
   - Validate file types/sizes for security.
3. **Node.js & Express**
   - Build a server to process form data and file uploads.
   - Connect Node.js to MongoDB for data storage.

4. **Data Validation & Security**
    - Sanitize inputs and prevent common vulnerabilities.
5. **Pagination & Advanced Queries**
    - Implement pagination in MongoDB results.
6. **Debugging & Error Handling**
    - Fix connection issues and handle errors gracefully.
7. **Full-Stack Basics**
    - Create simple backend systems with PHP/Node.js + MongoDB.

References:

1. **PHP**: php.net
2. **MongoDB**: docs.mongodb.com
3. **Node.js**: nodejs.org
4. **Express**: expressjs.com
5. **Multer**: GitHub
6. **OWASP Security**: owasp.org