PROJECT REPORT ON - WEB TRAINING PORTAL

(INDUSTRIAL TRAINING)

An

Industrial Training Report Submitted

On

08 July 2025

Indian Farmers Fertiliser Cooperative Limited

Phulpur Unit

Prayagraj U.P. 212404

For

The Fulfillment of Summer Internship

Guided by
Mr. Sandeep Goel
(IT services)
IT Department
IFFCO , Phulpur Unit

Submitted by
Rachit Pandey
Roll no. 2205916
CSE Department
KIIT , Bhubaneshwar, Odisha

ACKNOWLEDGMENT

I am very thankful to **IFFCO** for my summer internship. Special thanks to **Shri R.K. Pandey, Manager Training**, and **Shri Anubhav Verma, DGM IT Services**, at IFFCO Phulpur, Allahabad, for this opportunity.

I am also grateful to **Shri Sandeep Goel** for his invaluable supervision and support during my training. Thanks to all IFFCO Phulpur staff for their assistance and shared knowledge.

Finally, I appreciate my family and friends for their continuous support.

Rachit Pandey

Roll no. 2205916

CSE Department KIIT, Bhubaneshwar, Odisha

ABOUT IFFCO

Indian Farmers Fertiliser Cooperative Limited (IFFCO) is a global cooperative, fully owned by Indian cooperatives and founded in 1967. IFFCO provides quality **fertilizers** to boost India's agricultural output.

Starting with its first plant in Kalol, Gujarat, IFFCO has grown to operate five large fertilizer plants and several joint ventures. Its main products include **urea, DAP, NPK, and water-soluble fertilizers**.

Owned by over 36,000 cooperative societies, IFFCO's governance prioritizes transparency and farmer interests. It is also committed to sustainability, promoting **nano-fertilizers** and **organic farming**.

Future Prospects

IFFCO aims to expand globally, utilizing digital technologies and promoting sustainable farming practices. This ensures food security and ongoing support for Indian farmers.

In essence, IFFCO plays a vital role in empowering farmers and boosting agricultural productivity. This is achieved through its cooperative model, innovative products, and unwavering commitment to sustainability.

IT DEPARTMENT OF IFFCO

The **IT Department at Indian Farmers Fertiliser Cooperative Limited (IFFCO)** is crucial for its operations, leading digital transformation and innovation to support its vast network and boost agricultural productivity. It ensures smooth functioning from manufacturing to distribution.

Core Functions and Responsibilities

The IT Department develops, implements, and maintains IFFCO's technological infrastructure. Key responsibilities include:

- **System Integration:** Seamlessly integrating business processes via **ERP systems** for efficient resource, production, and supply chain management.
- **Data Management:** Overseeing data collection, storage, and analysis for actionable insights, including tracking inventory, production, and sales.

- **Network Security:** Implementing advanced cybersecurity measures, including regular updates, monitoring, and training, to protect sensitive information.
- **Application Development:** Creating and maintaining custom software for finance, HR, logistics, and CRM.
- **Technical Support:** Providing continuous support, troubleshooting, maintenance, and system upgrades.

Innovations and Digital Initiatives

The IT Department drives digital initiatives to modernize agricultural practices and improve farmer engagement:

- **IFFCO Kisan App:** A mobile app providing farmers with weather forecasts, crop advisories, and market prices.
- **E-commerce Platform:** An online platform for selling fertilizers and agricultural inputs directly to farmers.

Challenges and Solutions

The department tackles challenges like continuous innovation, data security, and system integration across multiple locations by:

- **Training and Development:** Regularly updating IT staff skills.
- **Collaborations:** Partnering with tech providers and research institutions.
- **Infrastructure Upgrades:** Continuously improving IT infrastructure for performance, reliability, and security.

Future Prospects

In summary, IFFCO's IT Department is fundamental to its operations, driving digital transformation and innovation. By leveraging technology, it supports IFFCO's mission to enhance agricultural productivity and empower farmers across India.

Project Report: IFFCO Training & Certification Portal

This report provides a detailed, in-depth analysis of the IFFCO Training & Certification Portal, examining the purpose, functionality, and key code aspects of each PHP file provided.

1. Project Overview

The IFFCO Training & Certification Portal is a web-based system designed to manage the lifecycle of training participants, from registration and project submission to administrative approval and certificate issuance. It implements a clear separation of concerns with distinct user and administrator interfaces and functionalities. The portal is built using PHP for server-side logic, MySQL for data storage, and Tailwind CSS for styling, with Chart.js integrated for data visualization in the admin section.

2. Core Project Structure

The project organizes its files into logical directories, enhancing maintainability and clarity:

```
htdocs/
|   control.php             (Admin Control Center)
|   db_connect.php          (Database Connection)
|   index.php               (Login / Entry Point)
|   logout.php              (Session Logout)
|
├──admin/
|       applications.php    (Manage Pending Applications)
|       dashboard.php       (Admin Dashboard)
|       students.php        (Student Management / Certificate Issuance)
|       view_docs.php       (View Student Uploaded Documents)
|
├──assets/                   (Static assets like CSS, images)
|   └──CSS/
|          tailwind.min.css
|
```

```
|
├──uploads/                  (System-wide uploads like logos, project results)
|   |   bg.png
|   |   IFFCOlogo.svg
|   |   stamp.png
|   |
|   └──projects/             (Admin-managed project result PDFs)
|          6859031a59324-result.pdf
|          6859037225936-result.pdf
|
└──user/                     (User-specific functionalities)
    |   certificates.php    (View/Download Certificates)
    |   dashboard.php       (User Dashboard)
    |   profile.php         (User Profile Management)
    |   projects.php        (Project Submission)
    |   register.php        (New User Registration)
    |
    └──uploads/             (User-specific uploads like referral PDFs, NOCs)
        └──25CE1/           (Example user-specific folder based on roll number)
               noc.pdf
               referral.pdf
```

3. Detailed File Analysis

3.1. Core Files (htdocs/)

- **index.php**
  - **Purpose:** This is the main entry point for the application, handling user login and routing. It displays the login form to unauthenticated users.
  - **Functionality:**
    - Starts a PHP session to manage user state.
    - Includes db_connect.php for database interaction.
    - Processes POST requests for login:
      - Validates email and password inputs.
      - Uses a prepared statement to query the users table, fetching id, name, password (hashed), role, and status.
      - Verifies the provided password against the stored hash using password_verify().
      - Checks if the user's status is 'approved'.
      - Upon successful and approved login, sets session variables (loggedin, id, name, role).
      - Redirects to admin/dashboard.php for 'admin' roles and user/dashboard.php for 'user' roles.
    - Displays error messages for invalid credentials or unapproved accounts.
    - Provides a link to user/register.php for new registrations.
  - **Dependencies:** db_connect.php.
  - **Security:** Uses prepared statements for login to prevent SQL injection. Passwords are hashed.

- **db_connect.php**
  - **Purpose:** Establishes and manages the connection to the MySQL database.
  - **Functionality:**
    - Defines database connection parameters: host ('localhost'), username ('root'), password (''), and database ('if0_38954512_user').
    - Uses mysqli_connect() to establish the connection.
    - Includes error handling to die() with a JSON error message if the connection fails.
    - Sets the character set to utf8mb4 for proper character encoding.
  - **Dependencies:** None (core utility).
  - **Security:** Basic error handling. Hardcoded credentials are a security risk in production environments.
- **logout.php**
  - **Purpose:** Handles user logout by terminating the session.

    - **Functionality:**
        - Starts the PHP session.
        - Unsets all session variables ($_SESSION = array();).
        - Destroys the session (session_destroy()).
        - Redirects the user to index.php.
    - **Dependencies:** None.
- **control.php**
    - **Purpose:** Appears to be an "Admin Control Center" or a specific admin view that integrates data visualization and potentially dynamic content loading.
    - **Functionality:**
        - Includes Tailwind CSS and Chart.js for UI and data visualization.
        - Contains JavaScript for tabbed navigation (statsTab, tablesTab) to switch between different views (e.g., statistics and tables).
        - Features AJAX calls (e.g., fetchStats(), fetchTableData()) to dynamically load data, indicating interactive dashboards or data grids.
    - **Dependencies:** Tailwind CSS CDN, Chart.js CDN.
    - **Security:** As a front-end component, direct security concerns are minimal, but backend data fetching needs to be secure.

3.2. User Module (htdocs/user/)

- **register.php**
    - **Purpose:** Allows new users to register for the portal.
    - **Functionality:**
        - Includes db_connect.php.
        - Handles POST requests for registration:
            - Collects personal details: name, email, password, contact.
            - Collects program/academic details: program, department, college, semester, duration.
            - Collects referral information: referral_type, referral_file (PDF upload).
            - Generates a unique roll_no based on department shortcode (e.g., 'CS', 'ME'), current year, and a sequential number.
            - Determines batch from the current year.
            - Hashes the password using password_hash() before storing it.
            - Uploads the referral_file to a user-specific directory (user/uploads/{roll_no}/).
            - Inserts user data into the users table with a default status of 'pending' and role of 'user'.
        - Includes client-side form validation attributes (required).
    - **Dependencies:** db_connect.php.

    - **Security:** Uses password_hash() for secure password storage. Utilizes prepared statements for database insertion. Basic file type validation (accept=".pdf") for uploads.
- **dashboard.php** (User Dashboard)
    - **Purpose:** The main dashboard for logged-in users, providing an overview and navigation to other sections.
    - **Functionality:**
        - Authenticates user session; redirects to index.php if not logged in or not a 'user' role.
        - Displays the user's name.
        - Provides quick links to:
            - profile.php
            - projects.php
            - certificates.php
        - Includes a section for "Announcements".
        - Implements a mobile menu toggle using JavaScript.
    - **Dependencies:** db_connect.php.
    - **Security:** Enforces session and role-based access.
- **profile.php**
    - **Purpose:** Allows users to view and update their profile information. Also allows admins to view student profiles.
    - **Functionality:**
        - Authenticates user session; allows both 'user' and 'admin' roles.
        - Admins can view specific student profiles by passing a user_id via GET parameter.
        - Users can only edit their own profile ($can_edit flag).
        - Fetches user details (name, email, contact_info, roll_no, program, department, college, semester, duration, referral_type, status) from the users table.
        - Handles POST requests to update name, email, and contact_info for the logged-in user, using a prepared statement.
    - **Dependencies:** db_connect.php.
    - **Security:** Role-based access control, prepared statements for updates.
- **projects.php**
    - **Purpose:** Enables users to upload, view, and manage their project files and reports.
    - **Functionality:**
        - Authenticates user session; restricts access to 'user' role.
        - Retrieves the user's roll_no to construct a unique upload directory (user/uploads/{roll_no}/).

        - Handles file uploads for:
            - Project ZIP file (project.zip).
            - Report PDF file (report.pdf).
            - Renames uploaded files to a standard name (e.g., project.zip, report.pdf) and moves them to the user's directory.
            - Updates the projects table with file paths and submission details.
        - Handles delete requests for report.pdf and project.zip, removing both the file from the filesystem and its path from the database.
        - Displays the current status of project submission (e.g., "Submitted," "Not Started").
        - Provides download links for uploaded files.
    - **Dependencies:** db_connect.php.
    - **Security:** Uses file_exists() and unlink() for file deletion. It uses basename() when handling file names to prevent directory traversal issues. The hardcoded filename (project.zip, report.pdf) means only one of each can be stored per user.
- **certificates.php**
    - **Purpose:** Allows users to view and download their issued certificates.
    - **Functionality:**
        - Authenticates user session; restricts access to 'user' role.
        - Fetches certificate_path, issue_date, and qr_code_path from the certificates table for the logged-in user.
        - Displays certificate details and provides a download link for the PDF.
        - If a qr_code_path exists, displays the QR code image for verification.
        - Shows a "Certificate Locked" message if no certificate is found.
    - **Dependencies:** db_connect.php.
    - **Security:** Role-based access control.

3.3. Admin Module (htdocs/admin/)

- **dashboard.php** (Admin Dashboard)
    - **Purpose:** Provides administrators with a high-level overview of the portal's statistics.
    - **Functionality:**
        - Authenticates admin session; redirects if not logged in or not an 'admin' role.
        - Fetches and displays key statistics:
            - Total approved students.
            - Total pending applications.
            - Student counts per department (e.g., CS, HR, MBA).

        - Provides "Quick Actions" links to students.php for adding new students and importing students (though import functionality isn't detailed in provided files).
    - **Dependencies:** db_connect.php.
    - **Security:** Enforces admin-only access. Uses prepared statements for department-specific counts.
- **applications.php**
    - **Purpose:** Manages pending user applications, allowing administrators to review and act on them.
    - **Functionality:**
        - Authenticates admin session.
        - Fetches the toggle_status (auto-approve) for the current admin.
        - Handles POST requests:
            - **Toggle Auto-Approve:** Updates the toggle_status for the admin's account in the users table.
            - **Approve/Reject Application:** Updates the status of a user's application in the users table to 'approved' or 'rejected'. For approval, it also updates the approved_by_admin and approved_date fields.
        - Retrieves and displays a list of all 'pending' user applications, showing details like name, email, roll number, department, and batch.
        - Provides "See Docs" link to view_docs.php for each application.
        - Includes "Approve" and "Reject" buttons for each pending application.
    - **Dependencies:** db_connect.php, view_docs.php.
    - **Security:** Admin-only access. Uses prepared statements for database updates. confirm() for reject action.
- **students.php**
    - **Purpose:** Comprehensive student management for administrators, including deletion of student records and certificate issuance.
    - **Functionality:**
        - Authenticates admin session.
        - Includes qrcode/qrlib.php for QR code generation.
        - Handles POST requests:
            - **Delete Student:**
                - Retrieves roll_no for the student.
                - Deletes the student's associated upload folder (../user/uploads/{roll_no}/) and its contents recursively using a custom deleteFolder function.
                - Deletes the student's record from the users table.
            - **Issue Certificate:**

        - Collects student_id and certificate_pdf file.
        - Uploads the certificate PDF to uploads/projects/.
        - Generates a QR code image (PNG) for certificate verification, storing it in uploads/projects/. The QR code likely embeds a verification URL.
        - Inserts or updates the certificates table with user_id, certificate_path, issue_date, and qr_code_path.
    - Fetches and displays a list of all approved student users.
    - Provides "Delete Student" and "Issue Certificate" actions for each student via modals.
    - Allows viewing student profiles via profile.php.
  - **Dependencies:** db_connect.php, qrcode/qrlib.php, profile.php.
  - **Security:** Admin-only access. Implements prepared statements for database operations. Includes a function for recursive folder deletion. File uploads for certificates.
- **view_docs.php**
  - **Purpose:** Allows administrators to view documents uploaded by a specific student.
  - **Functionality:**
    - Authenticates admin session.
    - Takes roll (roll number) as a GET parameter. Uses basename() to sanitize the input to prevent path traversal.
    - Constructs the expected upload directory path (../user/uploads/{roll_no}/).
    - Fetches basic student data from the users table.
    - Scans the student's upload directory for files.
    - Identifies common document types (e.g., 'ref_', 'noc_' prefixes) and assigns labels.
    - Displays a list of found documents with their names, labels, appropriate icons, and "Open" links.
    - Provides a "Back to Students" link.
  - **Dependencies:** db_connect.php.
  - **Security:** Admin-only access. Uses basename() to prevent directory traversal in the roll_no parameter. Checks is_dir() and is_file() before listing or accessing files.

4. Technical Stack

- **Backend:** PHP
- **Database:** MySQL (accessed via mysqli)
- **Frontend:** HTML, CSS (Tailwind CSS), JavaScript (Chart.js for visualizations, basic DOM manipulation for UI interactions)

- **Libraries/Frameworks:**
    - Tailwind CSS (via CDN)
    - Chart.js (via CDN)
    - Font Awesome (for icons)
    - qrlib.php (for QR code generation)

5. Security and Best Practices (Review)

The project demonstrates several good security practices:

- **Prepared Statements:** Widely used throughout for all database interactions (login, registration, updates, deletions), effectively preventing SQL injection.
- **Password Hashing:** Passwords are hashed using password_hash() and verified with password_verify(), making them secure against direct disclosure.
- **Role-Based Access Control:** Clear separation of user and admin roles with session-based access checks ($_SESSION['role']).
- **Session Management:** Standard PHP session handling (session_start(), session_destroy()).
- **Path Sanitization:** Use of basename() in view_docs.php helps mitigate directory traversal vulnerabilities when constructing file paths from user input.
- **Error Reporting (Development):** ini_set('display_errors', 1); error_reporting(E_ALL); is enabled in some files (projects.php, register.php, students.php), which is useful for debugging during development. It should be disabled in production.

6. Conclusion

The IFFCO Training & Certification Portal is a well-structured application that effectively manages student training and certification processes. It leverages common web technologies and incorporates fundamental security measures like prepared statements and password hashing. Addressing the identified areas for improvement, particularly regarding production security practices and comprehensive input validation, would further strengthen the application's robustness and security posture.

(INSPIRED BY IFFCO LOGO)

GitHub : https://github.com/rachit9876/iffco.portal