ELECTRICITY BILL GENERATION DOCUMENTATION

Modularization

Modularization is the process of dividing a software system into **independent, manageable, and reusable modules**, where each module performs a specific task.

MODULE SPECIFICATIONS

Module 1: index.html

- **Module Name:** Home / Login Page
- **Purpose:** Entry point of the Electricity Bill System
- **Input:** User credentials (login selection)
- **Preconditions:** Web server must be running
- **Logic:**
    1. Display homepage
    2. Provide links to login/register
- **Output:** Redirects user to respective dashboard or login page

Module 2: config.php

- **Module Name:** Database Configuration
- **Purpose:** Establish database connection
- **Input:** Database host, username, password, database name
- **Preconditions:** MySQL server should be active
- **Logic:**
    1. Define database credentials
    2. Create connection using MySQLi
    3. Handle connection errors
- **Output:** Active database connection object.

Module 3: register_user.php

- **Module Name:** User Registration

- **Purpose:** Register new electricity consumers
- **Input:** Username, meter number, password
- **Preconditions:** User should not already exist
- **Logic:**
    1. Accept user details from form
    2. Validate inputs
    3. Insert data into users table
- **Output:** Successful registration message

Module 4: admin_dashboard.php

- **Module Name:** Admin Dashboard
- **Purpose:** Manage system users and employees
- **Input:** Admin login credentials
- **Preconditions:** Admin must be authenticated
- **Logic:**
    1. Display admin controls
    2. View users and employees
    3. Manage system data
- **Output:** Admin management interface

Module 5: employee_dashboard.php

- **Module Name:** Employee Dashboard
- **Purpose:** Generate and manage electricity bills
- **Input:** User ID, units consumed
- **Preconditions:** Employee must be logged in
- **Logic:**
    1. Fetch user details
    2. Enter units consumed
    3. Redirect to bill generation
- **Output:** Bill generation request processed

Module 6: generate_bill.php

- **Module Name:** Bill Generation
- **Purpose:** Calculate electricity bill
- **Input:** Units consumed, tariff details
- **Preconditions:** Valid user ID exists
- **Logic:**
    1. Read units consumed
    2. Apply tariff slabs
    3. Calculate total amount
    4. Store bill in database
- **Output:** Generated bill stored successfully

Module 7: view_bill.php

- **Module Name:** View Bill
- **Purpose:** Display electricity bill to user
- **Input:** User ID
- **Preconditions:** Bill must already be generated
- **Logic:**
    1. Fetch bill details from database
    2. Display units, amount, and date
- **Output:** Bill details shown on screen

Module 8: user_dashboard.php

- **Module Name:** User Dashboard
- **Purpose:** Allow users to view their bills
- **Input:** User login credentials
- **Preconditions:** User must be authenticated
- **Logic:**
    1. Display user information
    2. Provide option to view bills
- **Output:** User billing information displayed

Module 9: database.sql

- **Module Name:** Database Schema
- **Purpose:** Define database structure
- **Input:** SQL commands
- **Preconditions:** MySQL installed
- **Logic:**
    1. Create database
    2. Create tables (users, bills, employees)
- **Output:** Database schema created successfully

Algorithm Name: Electricity Bill Calculation

- **Objective:**
- To calculate the electricity bill amount based on the number of units consumed using slab-wise tariff rates and store the bill details in the database.

Input:

- Number of units consumed

Output:

- Total electricity bill amount

Algorithm Steps:

1. **Start**
2. **Read** the number of units consumed by the consumer.
3. **Initialize** tariff rates rate1, rate2, and rate3 according to the electricity board rules.
4. **If** the number of units consumed is less than or equal to 100, **then**
    - Calculate
      Bill = units × rate1
5. **Else if** the number of units consumed is greater than 100 **and** less than or equal to 200, **then**
    - Calculate
      Bill = (100 × rate1) + (units − 100) × rate2
6. **Else**
    - Calculate
      Bill = (100 × rate1) + (100 × rate2) + (units − 200) × rate3

7. **Display** the total electricity bill amount to the user.
8. **Store** the calculated bill details in the database for future reference.
9. **Stop**

TEST PLAN

Function 2: sanitize_input($data):

Field	**Description**
Test ID	TC_CFG_03
Functionality	Sanitize user input
Input	<script>alert(1)</script>
Expected Output	Sanitized safe string
Actual Output	Input sanitized successfully
Test Result	Pass

Field	Description
Test ID	TC_CFG_04
Functionality	Remove SQL injection characters
Input	' OR 1=1 --
Expected Output	Escaped and safe string
Actual Output	Input escaped successfully
Test Result	Pass

Function 3: generateServiceNo($category_id):

Field	**Description**
Test ID	TC_CFG_05
Functionality	Generate unique service number

Field	Description
Input	Category ID = 1
Expected Output	10-digit unique service number
Actual Output	Unique service number generated
Test Result	Pass

Field	Description
Test ID	TC_CFG_06
Functionality	Prevent duplicate service numbers
Input	Existing service number
Expected Output	New unique service number
Actual Output	Duplicate avoided successfully
Test Result	Pass

Function 4: generateBillNo():

Field	Description
Test ID	TC_CFG_07
Functionality	Generate bill number when bills exist
Input	Existing bill records
Expected Output	Incremented bill number
Actual Output	Bill number generated correctly
Test Result	Pass

Field	Description
Test ID	TC_CFG_08
Functionality	Generate first bill number
Input	Empty bills table

Field	Description
Expected Output	BILL-YYYY-000001
Actual Output	First bill number generated
Test Result	Pass

Function 5: isAdminLoggedIn() :

Field	Description
Test ID	TC_CFG_09
Functionality	Check admin login status
Input	Admin session set
Expected Output	Returns true
Actual Output	True
Test Result	Pass

Function 6: isEmployeeLoggedIn() :

Field	Description
Test ID	TC_CFG_10
Functionality	Check employee login status
Input	Employee session set
Expected Output	Returns true
Actual Output	True
Test Result	Pass

Function 7: isUserLoggedIn() :

Field	Description
Test ID	TC_CFG_11
Functionality	Check user login status

Field	Description
Input	User session set
Expected Output	Returns true
Actual Output	True
Test Result	Pass

QUALITY CHARACTERISTICS

Usability (Easy to Use)

- Simple and intuitive user interface
- Separate dashboards for Admin, Employee, and User
- Clear navigation and readable forms
- Error messages guide the user properly

Example:
Users can view bills with a single click, and employees can generate bills without technical knowledge.

Efficiency (Optimal Use of Resources)

- Single database connection reused using config.php
- Optimized SQL queries reduce processing time
- Minimal memory usage by avoiding redundant data storage
- Server-side validation reduces unnecessary page reloads

Result:
Fast response time and low server load.

Reusability

- Database connection module reused across all PHP files
- Bill calculation logic reused wherever required
- Authentication logic reused for different roles

Benefit:
Same modules can be reused in other billing or utility-based applications

Interoperability

- Uses PHP, MySQL, HTML, CSS
- Can run on different platforms (Windows, Linux)
- Works on multiple browsers
- Can be extended to mobile or API-based systems
