MINI PROJECT REPORT

A Report Submitted
In Partial Fulfillment
for award of Bachelor of Technology

In
COMPUTER SCIENCE & ENGINEERING

By

AKHILESH KUMAR (Roll No. 2201330100028)
AKHILESH KUMAR (Roll No. 2201330100027)
PARTH BHARDWAJ (Roll No. 2201330100173)
KRISH (Roll No. 2201330100136)

Under the Supervision of
Dr. Surya Prakash
Assistant Professor, CSE
NIET

a

Computer Science & Engineering Department
School of Computer Science & and Information Technology
NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute)
Affiliated to
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

DECLARATION

We hereby declare that the work presented in this report entitled "**SYSTEM PAYROLL MANAGEMENT SYSTEM**", was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name	: Akhilesh Kumar
Roll Number	: 2201330100028
Name	: Akhilesh Kumar
Roll Number	: 2201330100027
Name	: Parth Bhardwaj
Roll Number	: 2201330100173
Name	: Krish
Roll Number	: 2201330100136

CERTIFICATE

Certified that Akhilesh Kumar (Roll No_1: 2201330100028), Akhilesh Kumar (Roll No_2: 2201330100027), Parth Bhardwaj (Roll No_3: 2201330100173), and Krish (Roll No_4:) have carried out the research work presented in this Project Report entitled "System Payroll Management System" in partial fulfilment of the requirements for the award of the Bachelor of Technology in Computer Science & Engineering from Dr. A.P.J. Abdul Kalam Technical University, Lucknow, under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Dr Surya Prakash

Assistant Professor, CSE

NIET Greater Noida

Date:

Signature

Mrs Kumud Saxena

HOD, CSE

Greater Noida NIET

ACKNOWLEDGEMENTS

We would like to express my gratitude towards Dr Surya Prakash for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Our thanks and appreciations to respected Mrs Kumud Mam for their motivation and support throughout.

ABSTRACT

The Salary and Payroll Management System (SPMS) is a modular, scalable software solution engineered to simplify and automate payroll operations for small to medium-sized organizations. Designed with a three-tier architecture, SPMS ensures a clean separation of concerns between the user interface, business logic, and data management, making the system both maintainable and extensible.

SPMS streamlines the entire payroll lifecycle, from employee onboarding to salary computation and payslip generation. It supports dynamic tax calculations based on customizable rules, reducing manual errors and compliance risks. The system also offers intuitive reporting tools, providing insights into payroll trends, employee compensation, and statutory deductions.

Key Features

- Three-tier architecture – Separation of presentation, application, and data layers ensures modular design and maintainability.
- Employee management – Centralized tracking of employee data with department and role association.
- Dynamic payroll processing – Automates salary computation with allowance, deduction, and tax integration.
- Smart tax engine – Calculates taxes based on configurable brackets and employee-specific parameters.
- Comprehensive reporting – Generates payroll summaries, audit trails, and compliance reports for HR and finance teams.

SPMS serves as a foundational platform with room for future enhancements in security, modularity, and feature depth, aligning well with evolving organizational needs.

TABLE OF CONTENTS

Table OF CONTENTS

CHAPTER 1

1. INTRODUCTION

1.1 Background

Payroll processing is a critical function for organizations of all sizes, involving complex calculations, regulatory compliance, and accurate record-keeping. Traditional manual payroll systems are time-consuming, error-prone, and resource-intensive. The Salary and Payroll Management System (SPMS) was developed to address these challenges by providing an automated, efficient, and accurate solution for payroll management.

1.2 Purpose and Scope

The primary purpose of SPMS is to simplify and automate the complex process of payroll management. The system's scope encompasses:

- Comprehensive employee information management
- Salary calculation with customizable components
- Dynamic tax calculation with configurable tax brackets
- Attendance tracking and leave management integration
- Multiple earnings and deductions processing
- Detailed payslip generation and reporting
- Role-based access control for system security

The system is designed to be user-friendly, secure, and adaptable to various organizational structures and payroll policies, making it suitable for implementation across different business contexts.

1.3 Target Users

The SPMS is designed to serve multiple stakeholder groups within an organization:

- System Administrators: Responsible for overall system management, user account creation, and system configuration
- HR Managers: Handle employee records, process payroll, and ensure compliance with labor regulations
- Finance Team: Oversee financial calculations, tax compliance, and reconciliation with accounting systems
- Regular Staff Users: Process routine payroll operations and maintain basic employee records

Each user group has specific requirements and access privileges within the system, implemented through a role-based security model.

CHAPTER 2

2. LITERATURE REVIEW

2.1 Existing Payroll Systems

Payroll systems have evolved significantly from manual ledger-based approaches to sophisticated digital solutions. Current market offerings range from basic payroll calculators to enterprise-level Human Resource Information Systems (HRIS) with integrated payroll modules. Common limitations in existing systems include:

- Inflexibility in adapting to organization-specific requirements
- Complexity requiring specialized training
- High cost of implementation and maintenance
- Limited integration capabilities with other business systems
- Challenges in adapting to changing tax regulations

The SPMS addresses these limitations through its modular design, customizable components, and dynamic tax calculation system.

2.2 Technological Considerations

Modern payroll systems leverage various technologies to enhance functionality, security, and user experience. Key technological considerations for payroll system development include:

- Database design for efficient data storage and retrieval
- Secure authentication and authorization mechanisms
- Responsive user interface for cross-device compatibility
- Calculation engines for complex payroll formulas
- Reporting and data visualization capabilities
- Integration interfaces with external systems

The SPMS implementation incorporates these considerations through its three-tier architecture, responsive Bootstrap-based UI, and modular code organization.

2.3 Regulatory Requirements

Payroll systems must comply with various regulatory requirements, including:

- Tax withholding regulations
- Labor laws regarding minimum wage and overtime
- Data privacy and protection regulations
- Financial reporting standards
- Record-keeping requirements

The SPMS addresses these requirements through its dynamic tax calculation system, configurable deductions, and comprehensive record-keeping capabilities.

CHAPTER 3

3. METHODOLOGY

3.1 Development Approach

The SPMS was developed using a combination of structured and object-oriented approaches:

- Requirements Analysis: Detailed analysis of stakeholder needs and system requirements
- System Design: Three-tier architecture design with clear separation of concerns
- Iterative Development: Progressive implementation of core modules with regular testing
- Documentation: Comprehensive documentation of code, database schema, and user interfaces

This approach ensured that the system met stakeholder requirements while maintaining code quality and system integrity.

3.2 Tools and Technologies

The SPMS implementation utilized the following tools and technologies:

- Frontend:
- HTML5 and CSS3 for markup and styling
- Bootstrap 5 for responsive UI components
- JavaScript and jQuery for client-side interactions
- DataTables for interactive data presentation
- Summernote for rich text editing
- FontAwesome for iconography

- Backend:
- PHP (both procedural and object-oriented)

- AJAX for asynchronous communication
- Database:
- MySQL for data storage
- SQL for data manipulation and retrieval

- Development Tools:
- Version control for code management
- Code editors and IDEs for development
- Browser developer tools for debugging

This technology stack was selected for its widespread adoption, robust community support, and suitability for web-based application development.

3.3 Implementation Strategy

The implementation strategy followed these key principles:

- Modular Development: Core system components were developed as separate modules
- Database-First Approach: Database schema was designed and implemented before application logic
- Progressive Enhancement: Basic functionality was implemented first, followed by advanced features
- Continuous Testing: Regular testing throughout the development process
- Documentation: Ongoing documentation of code, APIs, and user interfaces

This strategy ensured systematic development and integration of system components while maintaining code quality and system stability.

CHAPTER 4

4. SYSTEM DESIGN

4.1 Technical Architecture

The SPMS follows a traditional three-tier architecture with clear separation between presentation, application, and data layers:

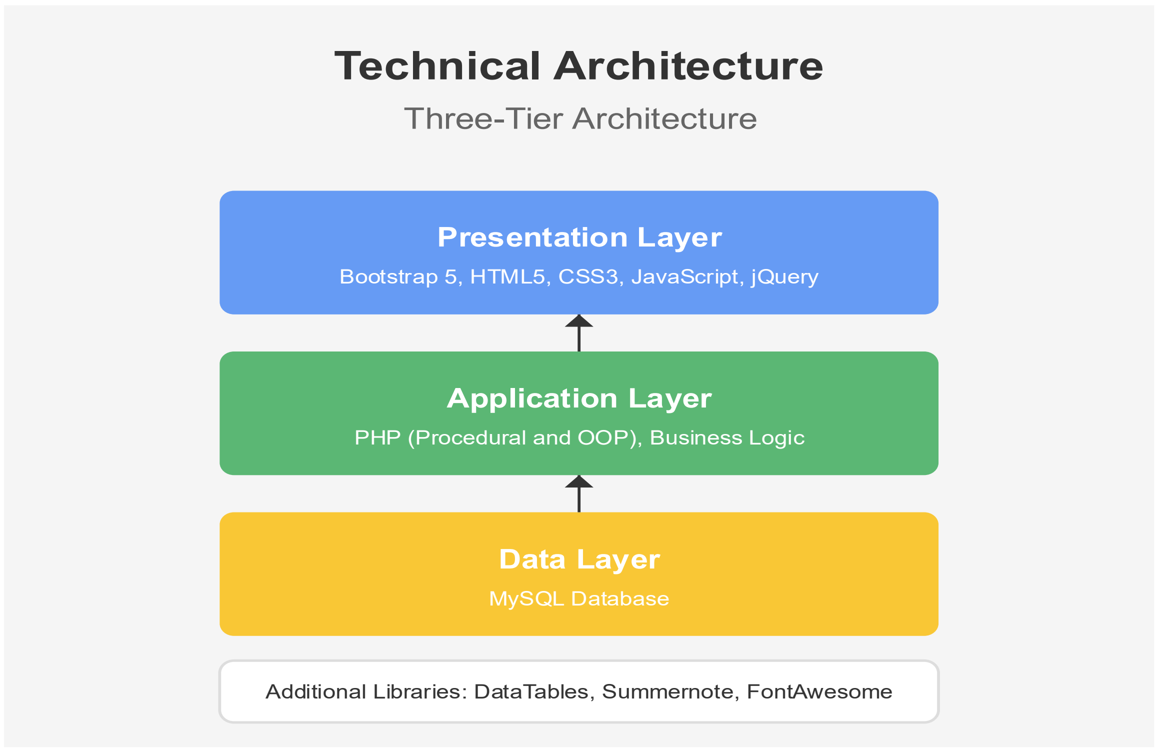

Technical Architecture
Three-Tier Architecture
Presentation Layer
Bootstrap 5, HTML5, CSS3, JavaScript, jQuery
Application Layer
PHP (Procedural and OOP), Business Logic
Data Layer
MySQL Database
Additional Libraries: DataTables, Summernote, FontAwesome

Fig.1 Technical Architecture

Presentation Layer

- Responsible for user interface rendering and client-side interactions
- Implemented using HTML5, CSS3, Bootstrap 5, and JavaScript/jQuery
- Features responsive design for cross-device compatibility
- Utilizes AJAX for asynchronous communication with the server

- Incorporates third-party libraries including DataTables for interactive data presentation, Summernote for rich text editing, and FontAwesome for iconography

Application Layer

- Handles business logic processing, data validation, and application flow
- Implemented in PHP with both procedural and object-oriented approaches
- Key components include:
- Actions.php`: Core business logic implementation with comprehensive payroll calculation methods
- DBConnection.php`: Database connection management and transaction handling
- Session management for authentication and state persistence
- Form processing with client and server-side validation
- AJAX request handlers for asynchronous operations

Data Layer

- Manages data storage, retrieval, and integrity
- Implemented using MySQL database with relational tables
- Features constraints and relationships for data integrity
- Supports transaction management for critical operations
- Implements efficient indexing for performance optimization

The communication flow follows a typical request-response cycle:

1. User interacts with the presentation layer through the browser

2. Client-side JavaScript makes AJAX calls to the server

3. PHP scripts process the request and perform business logic

4. PHP connects to MySQL for data operations

5. Results are formatted as JSON responses

6. Client-side JavaScript updates the UI based on the response

This architecture provides several advantages:

- Clear separation of concerns for maintainability

- Modular design allowing for component-level updates
- Scalability through distributed processing
- Enhanced security through layered access controls

4.2 Database Schema

The database schema is designed to support comprehensive payroll management with flexibility for various organizational structures and payroll policies. The relational database design ensures data integrity while supporting complex payroll operations.

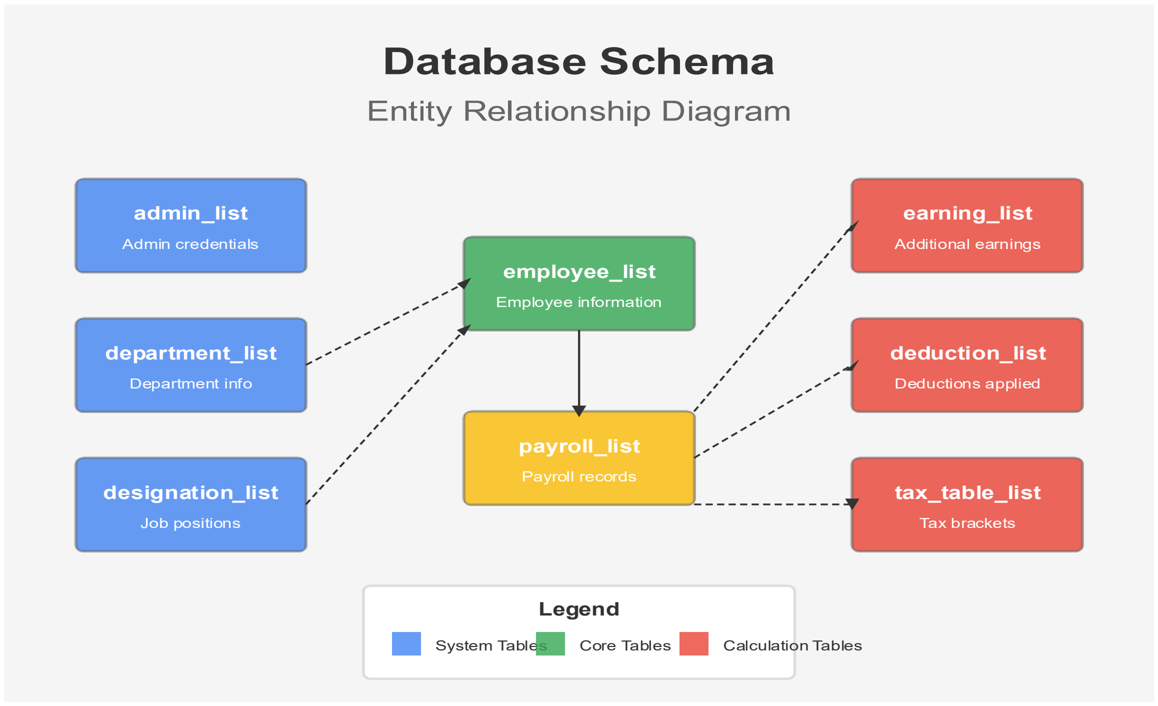


Fig.2 Database Schema

Key tables include:

User Management

- admin_list: Stores administrator credentials and information

- Primary key: `admin_id`
- Supports role-based access with `type` field
- Implements basic security with password hashing

Organizational Structure

- department_list: Manages department information
- Primary key: `department_id`
- Tracks department status (active/inactive)

- designation_list: Stores job positions/designations
- Primary key: `designation_id`
- Tracks designation status (active/inactive)

Employee Management

- employee_list: Contains comprehensive employee information
- Primary key: `employee_id`
- Foreign keys: `department_id`, `designation_id`
- Stores personal details, contact information, and salary information

Payroll Processing

- payroll_list: Stores payroll records for employees
- Primary key: `payroll_id`
- Foreign key: `employee_id`
- Stores calculation data including rates, attendance, and tax information

- earning_list: Manages additional earnings for payroll
- Foreign key: `payroll_id`
- Supports both taxable and non-taxable earnings

- deduction_list: Tracks deductions applied to payroll
- Foreign key: `payroll_id`
- Supports various deduction types

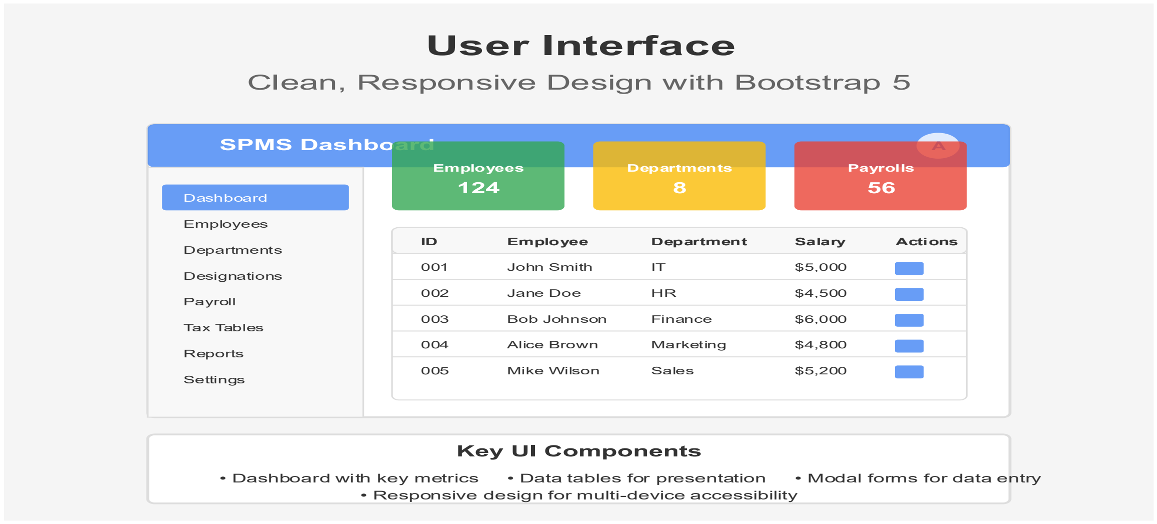
Tax Management

- tax_table_list: Stores tax brackets for withholding tax calculation
- Primary key: `tax_id`
- Supports different payroll types
- Implements effective dates for tax bracket changes

The database schema implements relationships between tables to maintain data integrity and support complex queries for payroll processing.

4.3 User Interface Design

The SPMS user interface is designed with a focus on usability, clarity, and efficiency. Key design principles include:



ID	Employee	Department	Salary	Actions
001	John Smith	IT	$5,000	🟦
002	Jane Doe	HR	$4,500	🟦
003	Bob Johnson	Finance	$6,000	🟦
004	Alice Brown	Marketing	$4,800	🟦
005	Mike Wilson	Sales	$5,200	🟦

Key UI Components

• Dashboard with key metrics • Data tables for presentation • Modal forms for data entry
• Responsive design for multi-device accessibility

FIG 3 User Interface

- Responsive Layout: Bootstrap-based responsive design for cross-device compatibility
- Intuitive Navigation: Clear menu structure and consistent navigation patterns
- Visual Hierarchy: Emphasis on important information through typography and color
- Interactive Elements: Dynamic forms and tables for efficient data entry and retrieval
- Feedback Mechanisms: Clear notifications and alerts for user actions
- Accessibility Considerations: High contrast text and scalable interface elements

Interface Organization

The interface is organized into logical sections corresponding to system modules:

- Dashboard: Overview of system status and key metrics
- Quick access to frequently used functions
- Summary statistics and notifications
- Recent activity tracking

- Employee Management: Forms and tables for employee data management
- Comprehensive employee information forms
- Searchable and sortable employee listings
- Employee profile views with complete information

- Department/Designation Management: Interfaces for organizational structure
- Department and designation creation/editing
- Status toggling for active/inactive items
- Hierarchical organization display

- Payroll Processing: Multi-step forms for payroll calculation and processing
- Employee selection with search functionality
- Dynamic calculation of rates and adjustments
- Interactive earnings and deductions management
- Real-time tax calculation

- Tax Management: Interfaces for tax bracket configuration
- Tax bracket creation and editing
- Effective date management
- Payroll type differentiation

- Reports: Interfaces for generating and viewing reports

- Payroll summaries and detailed reports
- Filtering and search capabilities
- Printable payslip generation

The user interface implements role-based access control, showing only relevant options to each user type, enhancing both security and usability by presenting a focused interface appropriate to each user's responsibilities.

CHAPTER 5

5. IMPLEMENTATION DETAILS

5.1 Core Modules

The SPMS consists of several core modules, each handling specific aspects of payroll management:

User Authentication and Access Control

- Implements secure login and session management
- Supports role-based access control (Administrator and Regular User)
- Manages user accounts and permissions

Employee Management

- Handles employee data entry, update, and retrieval
- Manages employee status (active/inactive)
- Supports employee code generation for unique identification
- Links employees to departments and designations

Department and Designation Management

- Manages organizational structure definition
- Supports activation/deactivation of departments and designations
- Provides interfaces for structure maintenance

Payroll Processing

- Implements comprehensive payroll calculation workflow
- Supports various payroll types (monthly/semi-monthly)
- Handles attendance-based adjustments and overtime
- Processes additional earnings and deductions
- Calculates withholding tax based on dynamic tax brackets

Reporting and Payslip Generation

- Generates detailed payslips for employees
- Provides reporting interfaces for payroll analysis
- Supports filtering and searching of payroll records

Each module is implemented with appropriate separation of concerns, following the three-tier architecture pattern.

5.2 Payroll Processing Workflow

The payroll processing workflow in SPMS follows a structured sequence of steps that ensures accurate and comprehensive payroll calculation:

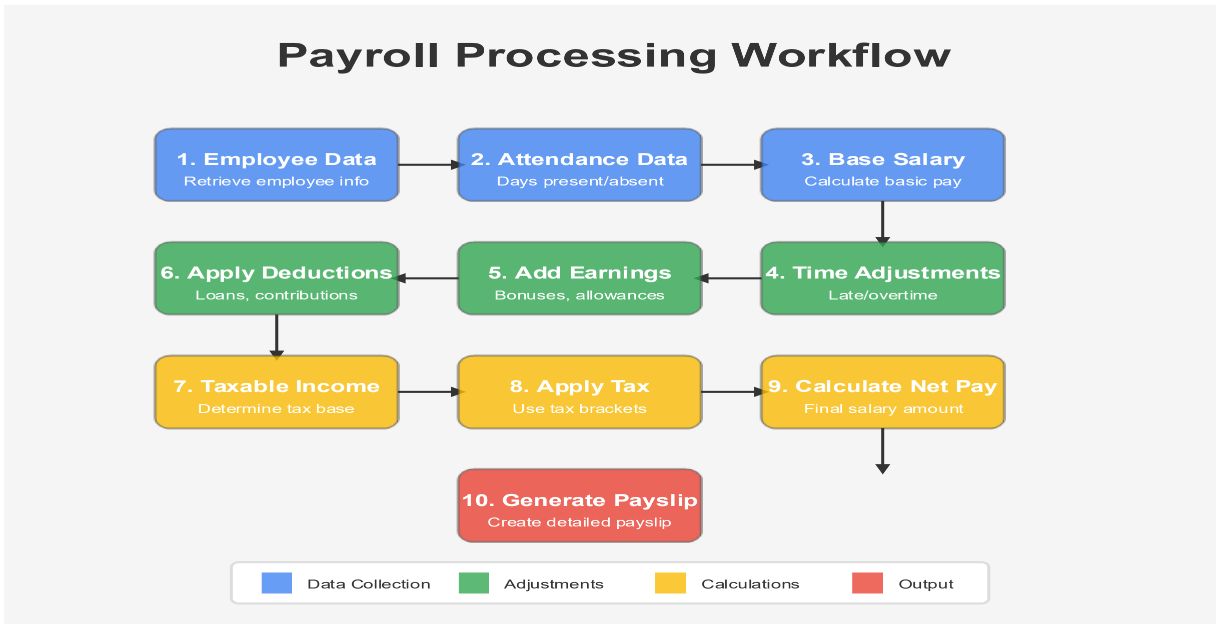


Fig 4 Payroll Processing Workflow

1. Employee Data Retrieval

- Retrieves employee's basic information and salary details from the database
- Calculates derived rates (daily, hourly, per-minute) based on standard formulas:
- Daily rate = Monthly salary ÷ 22 (standard working days)
- Hourly rate = Daily rate ÷ 8 (standard working hours)
- Per-minute rate = Hourly rate ÷ 60

- Implementation in `get_rate()` method of Actions class:

```php
function get_rate(){
extract($_POST);
$sql = "SELECT monthly_salary FROM `employee_list` where employee_id =
'{$employee_id}' ";
$result = $this->db->query($sql)->fetch_array();
if($result){
$resp'status'] ='success';
$salary = $result'monthly_salary'];
$resp'monthly'] = $salary;
$resp'daily'] = round($salary / 22,3);
$resp'hourly'] = round($resp'daily'] / 8,3);
$resp'per_minute'] = round($resp'hourly'] / 60,3);
}
return json_encode($resp);
}
```

2. Attendance and Time Adjustment Processing

- Processes present days, absences, late/undertime, and overtime
- Applies appropriate rates to calculate adjustments:
- Base salary = Daily rate × Number of present days
- Absence deduction = Daily rate × Number of absences
- Late/undertime deduction = Per-minute rate × Late/undertime minutes
- Overtime pay = Per-minute rate × Overtime minutes × Overtime rate
- Stored in database fields:

```sql
`no_present` double NOT NULL DEFAULT 0,
`no_absences` double NOT NULL DEFAULT 0,

`late_undertime` double NOT NULL DEFAULT 0,

`ot_min` double NOT NULL DEFAULT 0,
```

3. Additional Earnings Processing

- Handles multiple additional earnings per payroll
- Supports both taxable and non-taxable earnings
- Adds earnings to appropriate income category (taxable/non-taxable)
- Implementation for storing earnings:

```php
if(isset($earning_amount)){

$data = "";

foreach($earning_amount as $k => $v){

if(!empty($data)) $data.=", ";

$data .= "('{$payroll_id}','{$earning_name$k]}','{$v}','{$earning_tax$k]}')";

}

if(!empty($data)){

$this->db->query("INSERT INTO earning_list (`payroll_id`,`name`,`amount`,`taxable`)
VALUES {$data}");

}

}
```

4. Deduction Processing

- Processes various deduction types (loans, contributions, etc.)
- Subtracts deductions from gross pay
- Supports multiple deductions per payroll record
- Stored in separate `deduction_list` table for flexibility

5. Tax Calculation

- Determines taxable income by aggregating base salary and taxable earnings
- Selects appropriate tax bracket based on income, payroll type, and effective date
- Applies tax formula to calculate withholding tax
- Implements dynamic tax bracket selection:

```sql

SELECT fixed_tax, percentage_over FROM tax_table_list

WHERE '{amount}' BETWEEN range_from AND range_to

AND payroll_type = '{ptype}'

ORDER BY unix_timestamp(effective_date) DESC LIMIT 1

```

6. Net Pay Calculation

- Calculates net pay by subtracting tax and deductions from gross pay
- Adds non-taxable earnings to arrive at final net pay
- Formula: Net Pay = (Base Salary • Deductions • Tax) + Non-taxable Earnings

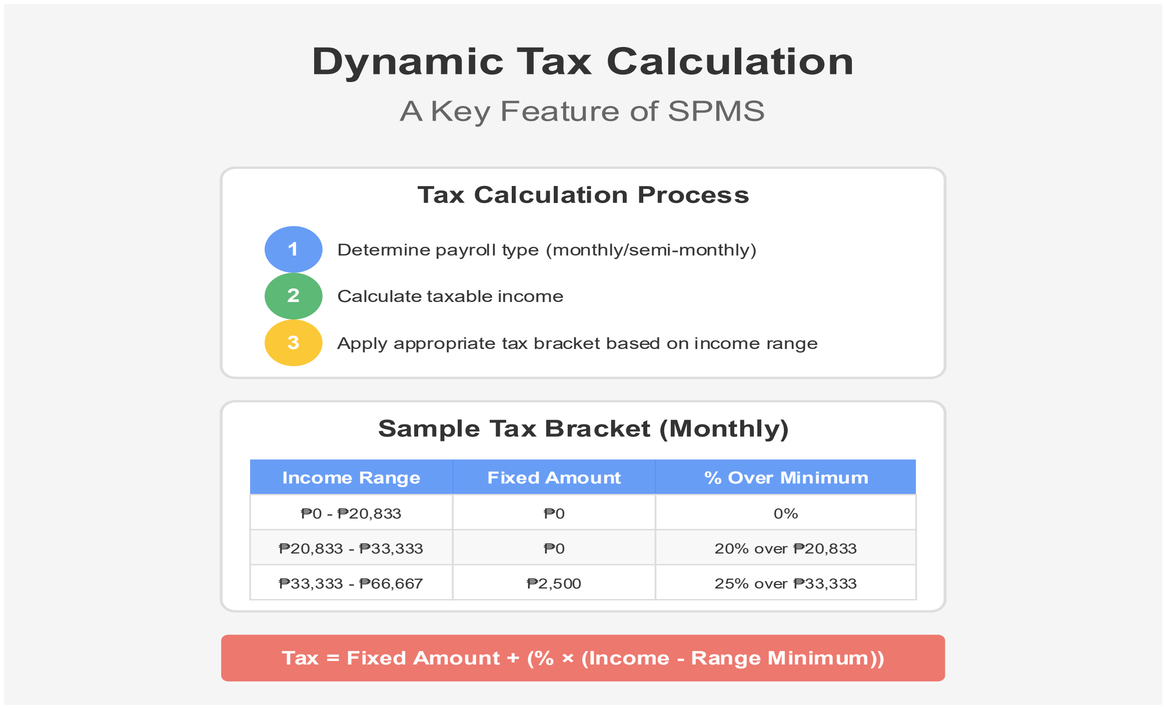
7. Payslip Generation

- Compiles all calculation results into a comprehensive payslip
- Formats payslip for presentation and printing
- Includes detailed breakdown of earnings, deductions, and tax

This workflow is implemented through a combination of server-side PHP code and client-side JavaScript, with AJAX communication providing a responsive user experience. The modular design allows for customization of each step to accommodate different organizational requirements and payroll policies.

5.3 Dynamic Tax Calculation System

One of the standout features of SPMS is its dynamic tax calculation system, which implements a sophisticated approach to withholding tax determination:


Dynamic Tax Calculation
A Key Feature of SPMS
Tax Calculation Process
1 Determine payroll type (monthly/semi-monthly)
2 Calculate taxable income
3 Apply appropriate tax bracket based on income range
Sample Tax Bracket (Monthly)
Tax = Fixed Amount + (% × (Income - Range Minimum))

Income Range	Fixed Amount	% Over Minimum
₱0 - ₱20,833	₱0	0%
₱20,833 - ₱33,333	₱0	20% over ₱20,833
₱33,333 - ₱66,667	₱2,500	25% over ₱33,333

Fig 5 Dynamic Tax Calculation

Taxable Income Determination

- Aggregates base salary and taxable additional earnings
- Adjusts for attendance and time factors
- Produces a consolidated taxable income figure
- Stored in database as `taxable_income` field

Tax Table Structure

The tax calculation relies on a well-structured tax table in the database:

```sql
```

```
CREATE TABLE `tax_table_list` (

`tax_id` int(30) NOT NULL,

`payroll_type` int(1) NOT NULL DEFAULT 1 COMMENT '1 = monthly, 2= semi-monthly',

`range_from` float NOT NULL DEFAULT 0,

`range_to` float NOT NULL DEFAULT 0,

`fixed_tax` float NOT NULL DEFAULT 0,

`percentage_over` float NOT NULL DEFAULT 0,

`effective_date` date NOT NULL,

`date_created` timestamp NOT NULL DEFAULT current_timestamp(),

`date_updated` timestamp NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()

)
```

This structure allows for:

- Different tax brackets for different income levels
- Different tax rates for monthly vs. semi-monthly payrolls
- Historical tracking of tax brackets through effective dates

Tax Bracket Selection

- Dynamically selects the appropriate tax bracket based on:
- Income amount (range matching)
- Payroll type (monthly/semi-monthly)
- Most recent effective date (for historical accuracy)
- Implements this selection through a SQL query:

```sql
SELECT fixed_tax, percentage_over FROM tax_table_list

WHERE '{amount}' BETWEEN range_from AND range_to

AND payroll_type = '{ptype}'
```

```
ORDER BY unix_timestamp(effective_date) DESC LIMIT 1
```

Tax Formula Application

- Applies a complex formula combining fixed tax and percentage components:

```php
function get_tax(){
extract($_POST);
$sql = "SELECT fixed_tax,percentage_over FROM `tax_table_list`
where '{$amount}' BETWEEN range_from and range_to
and payroll_type = '{$ptype}'
order by unix_timestamp(effective_date) desc limit 1";
$result = $this->db->query($sql)->fetch_array();
if($result){
$tax = $result'fixed_tax'];
if($result'percentage_over'] > 0){
$tax = $tax + ($amount * ($result'percentage_over'] / 100));
}
$resp'status'] = 'success';
$resp'tax'] = $tax;
}
return json_encode($resp);
}
```

- Returns calculated tax for inclusion in payroll processing

Advantages of This Approach

- Flexibility to accommodate different tax structures
- Ability to handle tax regulation changes without code modifications
- Support for different payroll types with appropriate tax brackets
- Historical accuracy through effective date tracking
- Automatic selection of the correct tax bracket without manual intervention

This approach provides several advantages:

- Flexibility to accommodate different tax structures
- Ability to handle tax regulation changes without code modifications
- Support for different payroll types with appropriate tax brackets
- Historical accuracy through effective date tracking

5.4 Code Organization

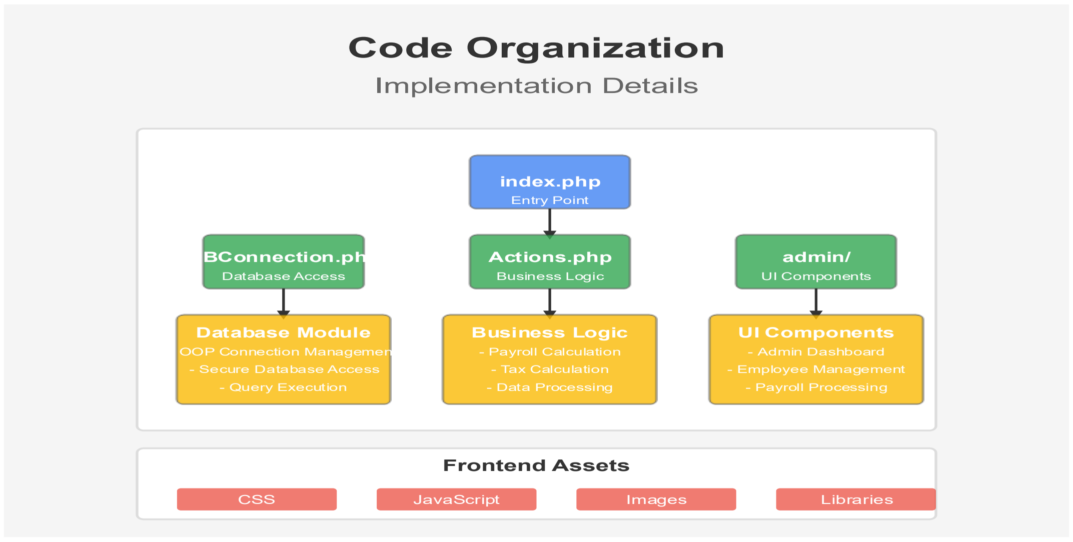


Fig 6 Implementation Details

The SPMS codebase is organized following a modular approach that enhances maintainability and separation of concerns:

File Structure

- Root Directory: Contains core system files and entry points
- `index.php`: Main entry point for the application
- `Actions.php`: Core business logic implementation
- `DBConnection.php`: Database connection management

- Admin Directory: Contains administrative interface files
- Authentication and dashboard files
- Module-specific management interfaces
- Reporting and configuration screens

- CSS/JS Directories: Frontend resources
- Bootstrap framework files
- Custom stylesheets and scripts
- Third-party libraries (DataTables, Summernote)
- Images Directory: Static image resources
- Upload Directory: User-uploaded content

Code Organization Principles

- Separation of Concerns: UI code is separated from business logic
- Modular Design: Each system function is implemented in dedicated files
- Consistent Naming: Predictable file and function naming conventions
- Reusable Components: Common functionality extracted into reusable functions
- Minimal Dependencies: Limited interdependencies between components

Implementation Patterns

- AJAX-Based Interaction: Client-server communication via JSON

```javascript
$.ajax({

url: _base_url_+"classes/Actions.php?a=save_payroll",

method: 'POST',
data: $(this).serialize(),
dataType: 'json',
error: err=>{
console.log(err)
alert_toast("An error occurred", 'error');
},
success:function(resp){
if(resp.status == 'success'){
location.replace('./?page=payroll/view_payroll&id='+resp.id);
}else{
alert_toast("An error occurred", 'error');
}
}
})
```

- Form Processing: Standardized approach to form submission and validation
- Database Interaction: Consistent query patterns and error handling
- UI Component Reuse: Common interface elements extracted into reusable code

This organization enhances maintainability, facilitates debugging, and supports future extensions to the system.

5.5 Security Implementation

The SPMS implements several security measures to protect sensitive payroll data:

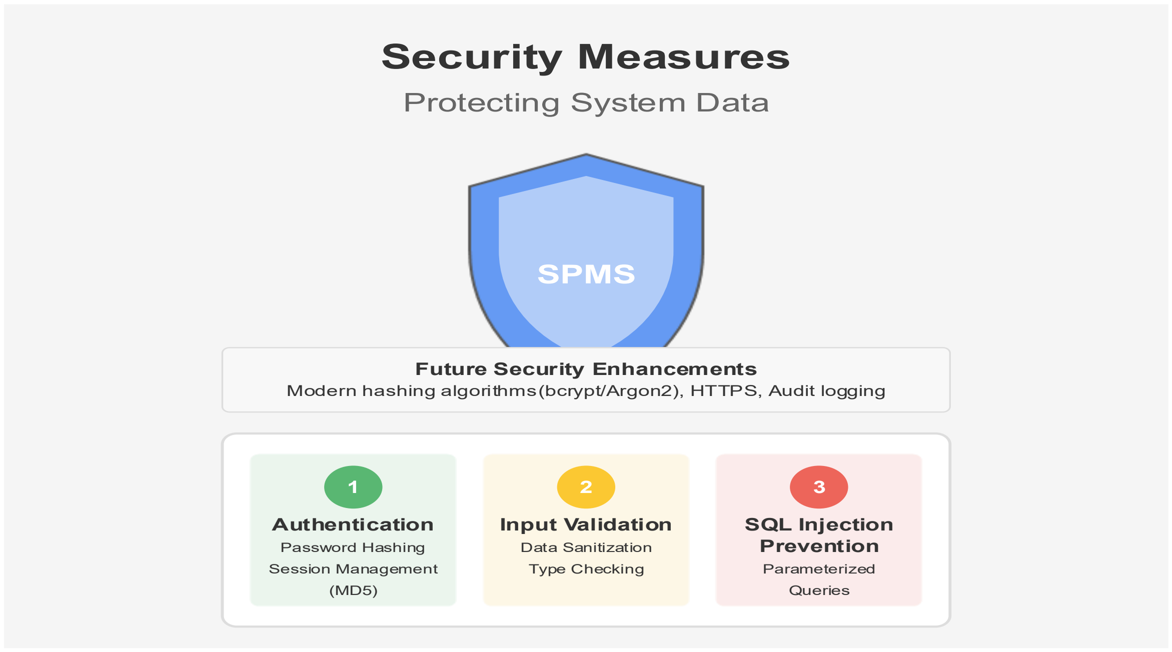


Fig 7 Security Measures

Authentication and Authorization

- Password-based authentication for system access
- Session management for maintaining authenticated state
- Role-based access control for feature restriction
- User type differentiation (Administrator vs. Regular User)
- Implementation in admin login process:

```php

function login(){

extract($_POST);

$sql = "SELECT * FROM admin_list where username = '{$username}' and password =
md5('{$password}') ";

$qry = $this->db->query($sql);

if($qry->num_rows > 0){

$res = $qry->fetch_array();

if($res'status'] != 1){

return json_encode(array('status'=>'notverified'));
}

foreach($res as $k => $v){

if(!is_numeric($k))

$_SESSION$k] = $v;

}

return json_encode(array('status'=>'success'));

}else{

return json_encode(array('status'=>'incorrect'));

}

}
```

Data Protection

- Password hashing using MD5 for credential storage
- Input validation to prevent SQL injection attacks
- Session timeout for inactive users
- Prepared statements for database queries
- Data sanitization for user inputs

Audit and Logging

- Timestamp tracking for critical operations
- User action logging for accountability
- Record of changes to sensitive data
- Creation and update timestamps on all records

Access Control Implementation

- Menu-based access restriction based on user role
- Session verification for protected pages
- Redirection of unauthorized access attempts

While these measures provide basic security, there are opportunities for enhancement in future versions of the system, including:

- Implementing stronger password hashing algorithms (bcrypt/Argon2)
- Adding two-factor authentication
- Enhancing audit logging with more detailed user activity tracking
- Implementing HTTPS for secure data transmission

CHAPTER 6

6. TESTING PROCEDURES

6.1 Unit Testing

Unit testing focused on verifying the correct functioning of individual components and functions within the system:

- Database Functions: Testing of CRUD operations for each table
- Calculation Functions: Verification of payroll calculation algorithms
- Validation Functions: Testing of input validation routines
- Authentication Functions: Verification of login and session management

Test cases were designed to cover both normal operation and edge cases, ensuring robust component behavior.

6.2 Integration Testing

Integration testing examined the interaction between system components:

- Module Integration: Testing of interactions between core modules
- Database Integration: Verification of complex queries and transactions
- UI-Backend Integration: Testing of form submissions and AJAX requests
- Workflow Integration: End-to-end testing of payroll processing workflow

These tests ensured that components worked together correctly to deliver system functionality.

6.3 User Acceptance Testing

User acceptance testing involved stakeholders evaluating the system against requirements:

- Functional Testing: Verification that system features met requirements
- Usability Testing: Evaluation of user interface and experience
- Performance Testing: Assessment of system responsiveness and efficiency
- Scenario Testing: Simulation of real-world payroll processing scenarios

Feedback from user acceptance testing informed final adjustments to the system before deployment.

CHAPTER 7

7. RESULTS AND DISCUSSION

7.1 System Performance

The SPMS demonstrates strong performance characteristics:

- Calculation Accuracy: Precise payroll calculations matching manual verification
- Processing Efficiency: Rapid processing of payroll data, even for large employee sets
- Response Time: Quick system response for user interactions
- Resource Utilization: Efficient use of server resources

Performance metrics indicate that the system is well-optimized for its intended use case, with room for scaling to larger organizations.

7.2 User Feedback

Feedback from system users has been predominantly positive:

- Usability: Users report intuitive navigation and clear workflows
- Functionality: The system meets core requirements for payroll processing
- Efficiency: Users note significant time savings compared to manual processes
- Reliability: Consistent system operation with minimal errors

Areas for improvement identified through user feedback include enhanced reporting capabilities and additional customization options.

7.3 Challenges Encountered

Several challenges were encountered during system development and implementation:

- Complex Tax Regulations: Implementing flexible tax calculation to accommodate changing regulations
- Diverse Payroll Scenarios: Supporting various payroll types and special cases
- Data Migration: Transferring existing payroll data into the new system
- User Training: Ensuring effective system adoption by users

These challenges were addressed through careful design, flexible implementation, and comprehensive documentation and training.

CHAPTER 8

8. DEPLOYMENT PROCESS

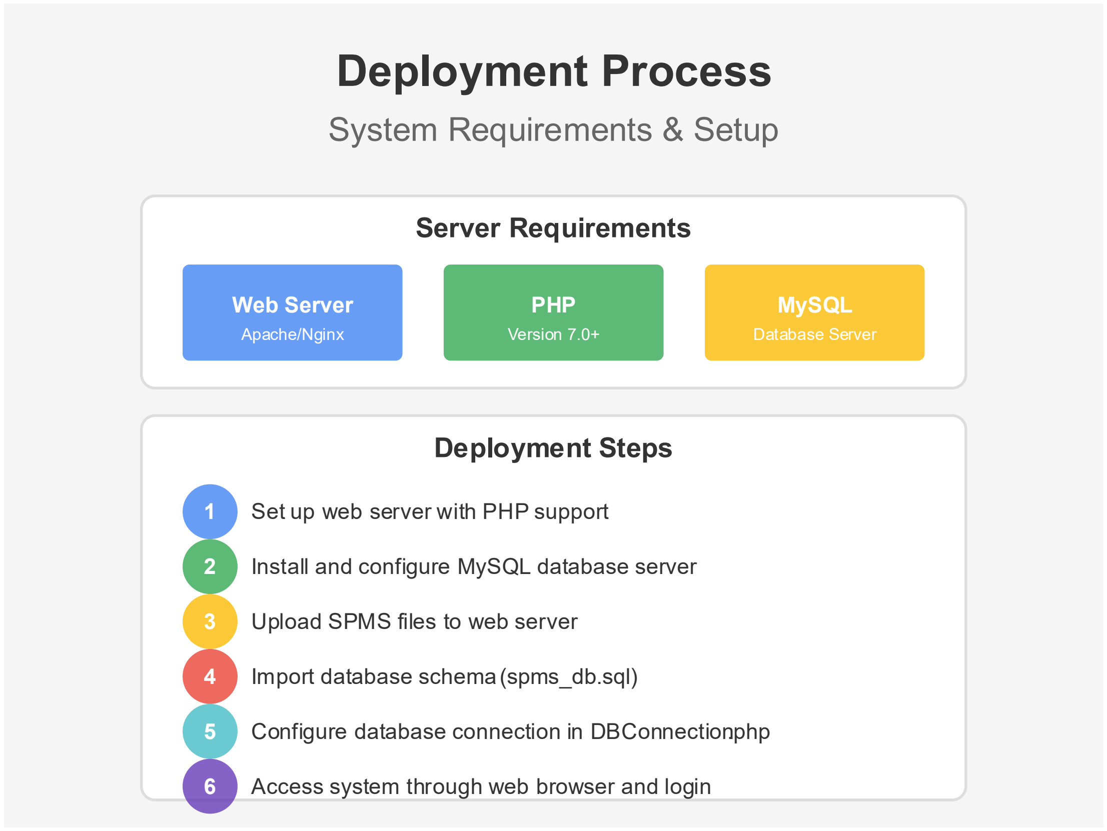

Deployment Process
System Requirements & Setup
Server Requirements
Web Server
Apache/Nginx
PHP
Version 7.0+
MySQL
Database Server
Deployment Steps
1 Set up web server with PHP support
2 Install and configure MySQL database server
3 Upload SPMS files to web server
4 Import database schema (spms_db.sql)
5 Configure database connection in DBConnectionphp
6 Access system through web browser and login

Fig 8 Deployment Process

The deployment of SPMS follows a structured process designed to ensure proper system setup and configuration:

8.1 Server Requirements

- Web Server: Apache/Nginx with PHP support
- PHP Version: PHP 7.2 or higher with required extensions:
- mysqli
- json
- session
- gd (for image processing)

- Database: MySQL 5.7 or higher
- Disk Space: Minimum 100MB for application files (excluding database)
- Memory: Minimum 512MB RAM recommended

8.2 Installation Steps

1. Server Preparation:

- Configure web server with appropriate PHP settings
- Create MySQL database and user with appropriate permissions

2. Application Deployment:

- Upload application files to web server document root
- Set appropriate file permissions
- Configure database connection in `DBConnection.php`

3. Database Setup:

- Import database schema from `db/spms_db.sql`
- Verify database tables and initial data

4. System Configuration:

- Create administrator account
- Configure system parameters
- Set up departments and designations

5. Testing and Verification:

- Verify all system functions
- Test payroll calculation with sample data
- Validate security measures

8.3 Maintenance Procedures

- Backup Strategy:
- Regular database backups
- Application file backups
- Configuration file preservation

- Update Process:
- Database schema updates
- Application file updates
- Configuration adjustments

- Performance Monitoring:
- Database query optimization
- Server resource monitoring
- Application response time tracking

Following this structured deployment process ensures proper system setup and reliable operation in production environments.

CHAPTER 9

9. LIMITATIONS AND FUTURE ENHANCEMENTS

9.1 Current Limitations

The current version of SPMS has several limitations that could be addressed in future updates:

Security Limitations

- Basic password hashing (MD5) rather than more secure algorithms
- Limited protection against CSRF attacks
- Minimal input validation in some areas

Functional Limitations

- No employee self-service portal
- Limited reporting and analytics capabilities
- No direct integration with accounting systems
- Limited support for complex organizational structures

Technical Limitations

- Limited optimization for large datasets
- No caching for frequently accessed data
- Limited mobile optimization

9.2 Proposed Enhancements

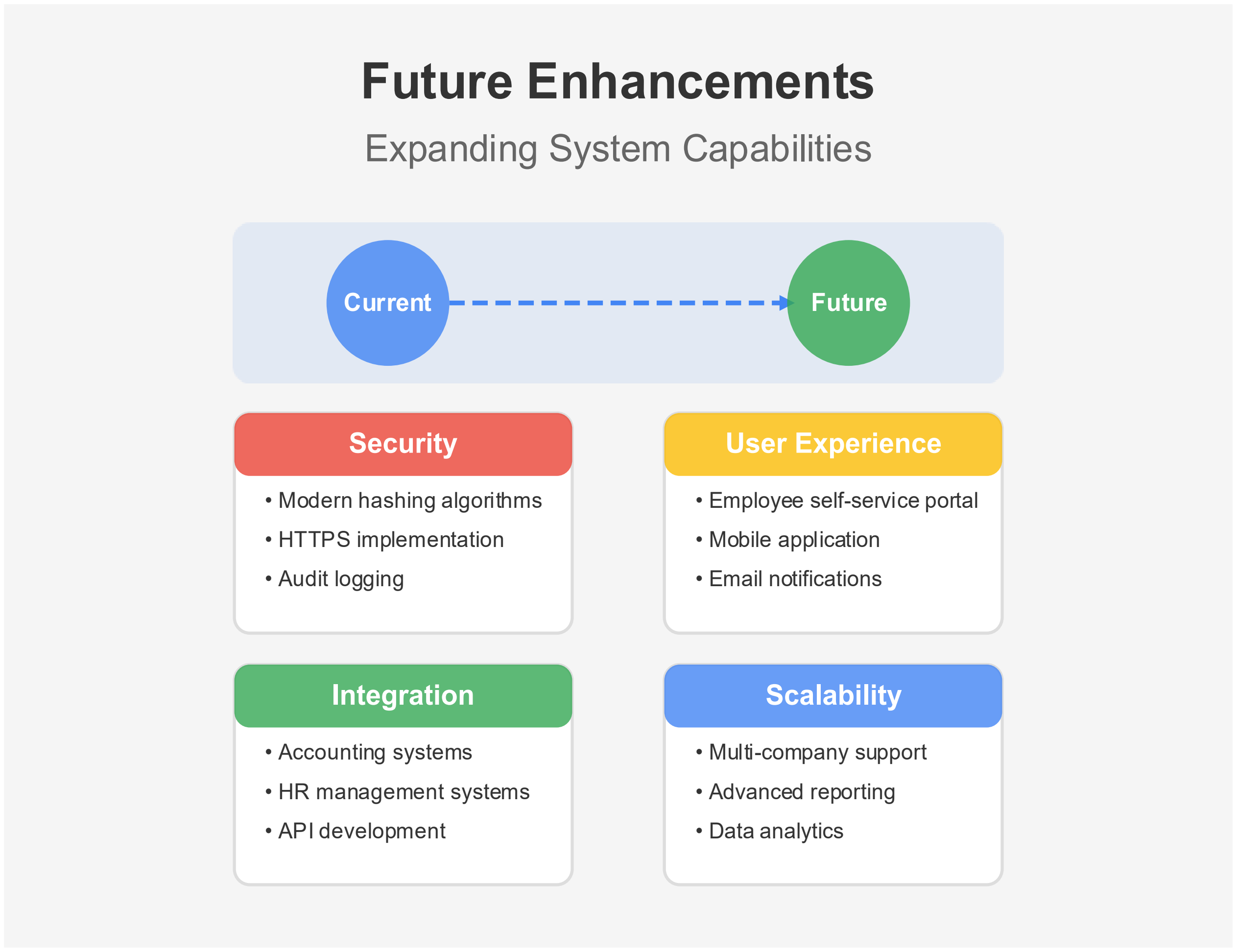


Fig 9 Future Enhancements

Based on identified limitations and user feedback, several enhancements are proposed for future versions:

Security Improvements

- Upgrade password hashing to more secure algorithms (bcrypt/Argon2)
- Implement CSRF protection for all form submissions

- Add comprehensive input validation and output encoding
- Implement more granular access control
- Add two-factor authentication for administrative access
- Implement encrypted data storage for sensitive information

Feature Enhancements

- Develop employee self-service portal
- Personal information management
- Leave request submission
- Payslip access and download
- Tax document retrieval
- Expand reporting and analytics capabilities
- Customizable report templates
- Data visualization dashboards
- Export options (PDF, Excel, CSV)
- Add integration with accounting systems
- API-based integration with popular accounting software
- Automated journal entry generation
- Reconciliation tools
- Support for more complex organizational structures
- Multi-level department hierarchies
- Matrix management relationships
- Project-based team assignments
- Implement document management for payroll-related documents

Technical Optimizations

- Query optimization for large datasets
- Indexed views for common queries
- Stored procedures for complex operations
- Query caching mechanisms
- Implement caching for frequently accessed data
- Enhance mobile experience with responsive design improvements
- Optimize asset delivery (minification, bundling)
- Implement progressive web app capabilities for offline access

These enhancements would address current limitations while expanding system capabilities to meet evolving organizational needs and technological advancements. The modular architecture of the current system provides a solid foundation for implementing these improvements incrementally.

CHAPTER 10

10. CONCLUSION

The Salary and Payroll Management System (SPMS) represents a comprehensive solution for automating and streamlining payroll processing and management. Through its modular design, three-tier architecture, and dynamic tax calculation system, SPMS provides organizations with a flexible, efficient tool for managing one of their most critical business processes.

Key achievements of the system include:

- Comprehensive Functionality: The system covers all essential aspects of payroll management, from employee data maintenance to payslip generation.
- Flexible Design: The modular architecture and configurable components allow adaptation to various organizational needs.
- Dynamic Tax Calculation: The sophisticated tax calculation system accommodates complex tax regulations and changes without code modifications.
- User-Friendly Interface: The responsive, intuitive interface enhances user productivity and reduces training requirements.

While the current implementation has some limitations, particularly in security and advanced features, these can be addressed in future enhancements. The solid foundation established by the current system provides a platform for ongoing development and improvement.

In conclusion, SPMS successfully meets its primary objective of simplifying and automating payroll management, offering significant benefits in terms of accuracy, efficiency, and compliance compared to manual or less sophisticated systems.

CHAPTER 11

11. REFERENCES

1. PHP Documentation. (2023). PHP Manual. https://www.php.net/manual/en/

2. MySQL Documentation. (2023). MySQL Reference Manual. https://dev.mysql.com/doc/

3. Bootstrap Documentation. (2023). Bootstrap 5 Documentation. https://getbootstrap.com/docs/5.0/

4. jQuery Documentation. (2023). jQuery API Documentation. https://api.jquery.com/

5. DataTables Documentation. (2023). DataTables Manual. https://datatables.net/manual/

6. Summernote Documentation. (2023). Summernote Documentation. https://summernote.org/

7. FontAwesome Documentation. (2023). FontAwesome Documentation. https://fontawesome.com/docs

8. OWASP. (2023). OWASP Top Ten. https://owasp.org/www-project-top-ten/

9. W3C. (2023). Web Content Accessibility Guidelines (WCAG). https://www.w3.org/WAI/standards-guidelines/wcag/

10. Internal Project Documentation. (2023). SPMS Technical Architecture Analysis.

11. Internal Project Documentation. (2023). SPMS Database Schema Analysis.

12. Internal Project Documentation. (2023). SPMS Payroll Processing Workflow.

13. Internal Project Documentation. (2023). SPMS Dynamic Tax Calculation System.

14. Internal Project Documentation. (2023). SPMS Stakeholder Analysis and Requirements.

CHAPTER 12

12. APPENDICES

Appendix A: System Diagrams

A.1 System Overview Diagram

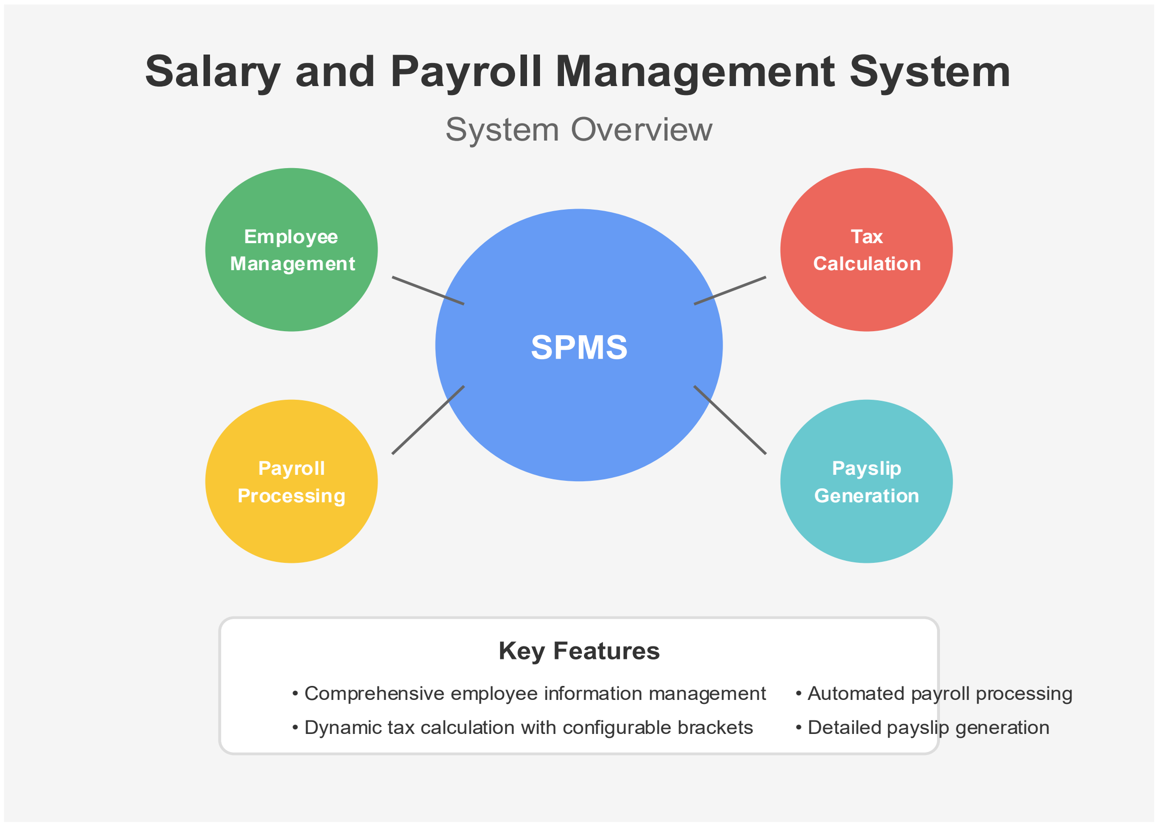

Salary and Payroll Management System
System Overview
Employee Management
Tax Calculation
SPMS
Payroll Processing
Payslip Generation
Key Features
• Comprehensive employee information management
• Automated payroll processing
• Dynamic tax calculation with configurable brackets
• Detailed payslip generation

Fig 10 Salary and Payroll Management System

A.2 Database Schema Diagram

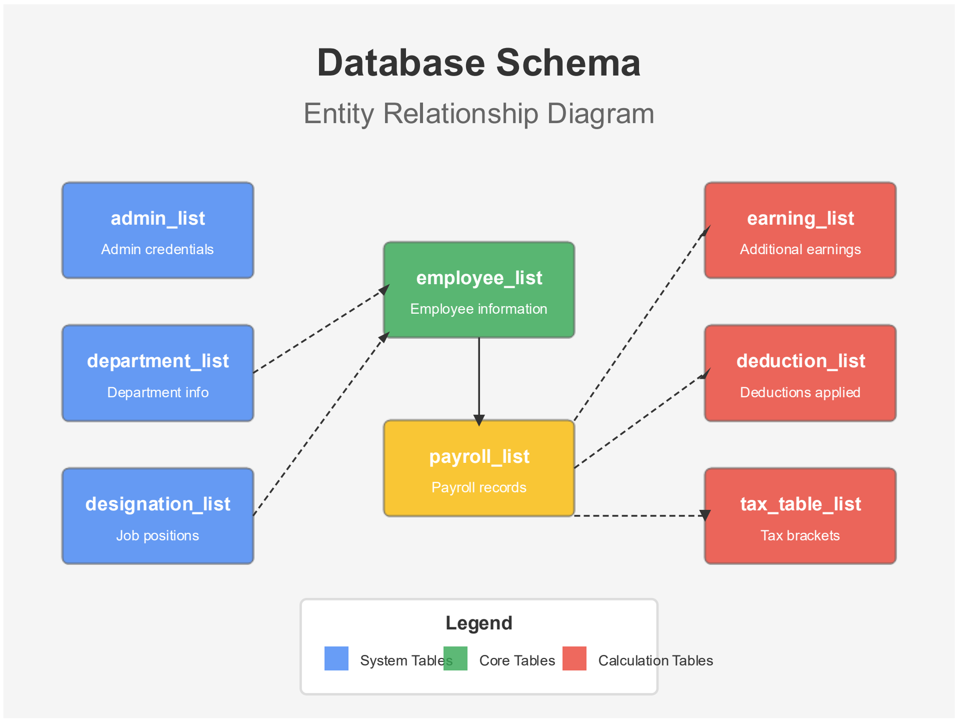


Fig 11 Database Schema

A.3 Payroll Processing Workflow Diagram

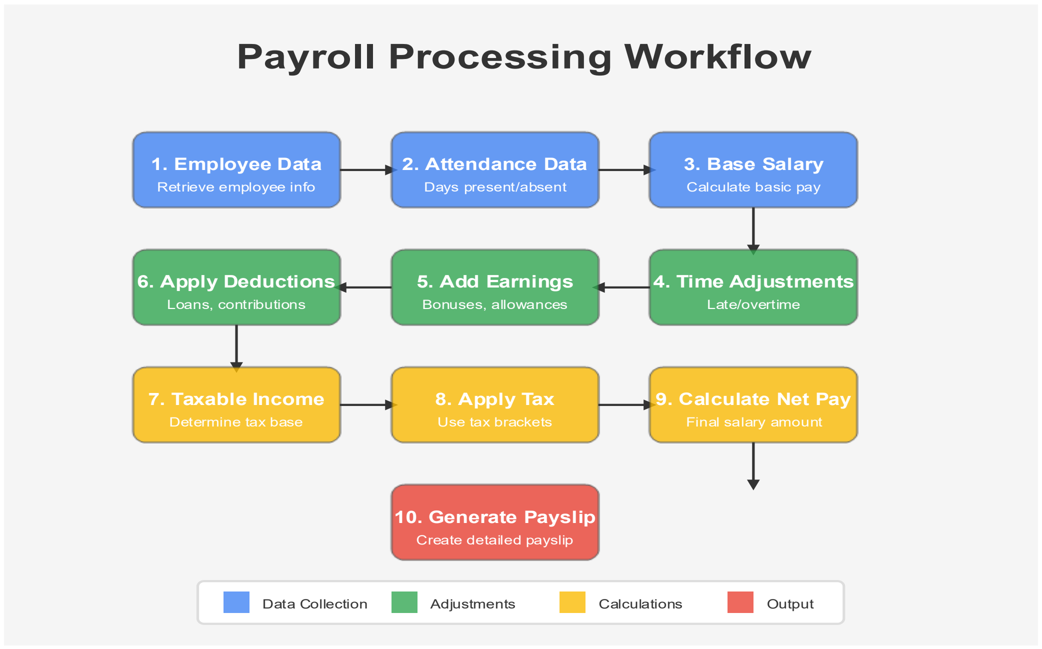
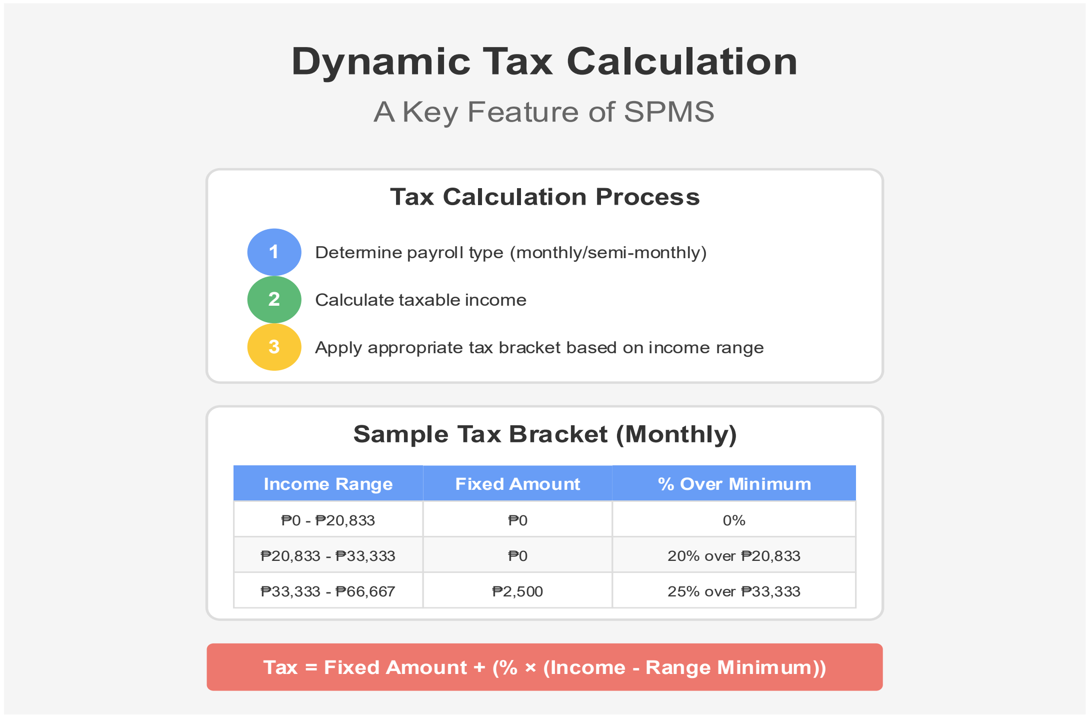


Fig 12 Payroll Processing Workflow

A.4 Tax Calculation Process Diagram


Dynamic Tax Calculation

A Key Feature of SPMS

Tax Calculation Process

1. Determine payroll type (monthly/semi-monthly)
2. Calculate taxable income
3. Apply appropriate tax bracket based on income range

Sample Tax Bracket (Monthly)

Income Range	Fixed Amount	% Over Minimum
₱0 - ₱20,833	₱0	0%
₱20,833 - ₱33,333	₱0	20% over ₱20,833
₱33,333 - ₱66,667	₱2,500	25% over ₱33,333

Tax = Fixed Amount + (% × (Income - Range Minimum))

Fig 12 Dynamic Tax Calculation

Appendix B: Code Samples

B.1 Tax Calculation Function

```php
function get_tax(){
extract($_POST);
$sql = "SELECT fixed_tax,percentage_over FROM `tax_table_list`
where '{$amount}' BETWEEN range_from and range_to
and payroll_type = '{$ptype}'
order by unix_timestamp(effective_date) desc limit 1";
$result = $this->db->query($sql)->fetch_array();
if($result){

$tax = $result'fixed_tax'];

if($result'percentage_over'] > 0){

$tax = $tax + ($amount * ($result'percentage_over'] / 100));

}

$resp'status'] = 'success';

$resp'tax'] = $tax;

}

return json_encode($resp);

}
```

B.2 Employee Rate Calculation Function

```php
function get_rate(){

extract($_POST);

$sql = "SELECT monthly_salary FROM `employee_list` where employee_id =
'{$employee_id}' ";

$result = $this->db->query($sql)->fetch_array();

if($result){

$resp'status'] ='success';

$salary = $result'monthly_salary'];

$resp'monthly'] = $salary;

$resp'daily'] = round($salary / 22,3);

$resp'hourly'] = round($resp'daily'] / 8,3);

$resp'per_minute'] = round($resp'hourly'] / 60,3);
```

```
}
return json_encode($resp);
}
```

Appendix C: Database Schema SQL

```sql
CREATE TABLE `admin_list` (
`admin_id` int(11) NOT NULL,
`fullname` text NOT NULL,
`username` text NOT NULL,
`password` text NOT NULL,
`type` int(11) NOT NULL DEFAULT 1,
`status` int(11) NOT NULL DEFAULT 1,
`date_created` timestamp NOT NULL DEFAULT current_timestamp()
);

CREATE TABLE `department_list` (
`department_id` int(11) NOT NULL,
`name` text NOT NULL,
`status` text NOT NULL DEFAULT 0,
`date_created` timestamp NOT NULL DEFAULT current_timestamp()
);

CREATE TABLE `designation_list` (
`designation_id` int(11) NOT NULL,

`name` text NOT NULL,

`status` text NOT NULL DEFAULT 0,

`date_created` timestamp NOT NULL DEFAULT current_timestamp()

);


CREATE TABLE `employee_list` (

`employee_id` int(11) NOT NULL,

`code` text NOT NULL,

`firstname` text NOT NULL,

`lastname` text NOT NULL,

`middlename` text DEFAULT NULL,

`gender` text NOT NULL,

`dob` text NOT NULL,

`email` text NOT NULL,

`contact` text NOT NULL,

`address` text NOT NULL,

`department_id` int(11) NOT NULL,

`designation_id` int(11) NOT NULL,

`monthly_salary` float NOT NULL DEFAULT 0,

`status` tinyint(1) NOT NULL DEFAULT 1,

`date_created` timestamp NOT NULL DEFAULT current_timestamp(),

`date_updated` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp()

);


CREATE TABLE `payroll_list` (
`payroll_id` int(11) NOT NULL,
`employee_id` int(11) NOT NULL,
`payroll_type` tinyint(1) NOT NULL DEFAULT 1 COMMENT '1 = monthly, 2= semi-monthly',
`date_from` date NOT NULL,
`date_to` date NOT NULL,
`monthly` float NOT NULL DEFAULT 0,
`daily` float NOT NULL DEFAULT 0,
`hourly` float NOT NULL DEFAULT 0,
`per_minute` float NOT NULL DEFAULT 0,
`no_present` double NOT NULL DEFAULT 0,
`no_absences` double NOT NULL DEFAULT 0,
`late_undertime` double NOT NULL DEFAULT 0,
`ot_min` double NOT NULL DEFAULT 0,
`taxable_income` double NOT NULL DEFAULT 0,
`nontaxable_income` double NOT NULL DEFAULT 0,
`tax` double NOT NULL DEFAULT 0,
`net` double NOT NULL DEFAULT 0,
`date_created` timestamp NOT NULL DEFAULT current_timestamp(),
`date_updated` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp()
);

CREATE TABLE `earning_list` (
`earning_id` int(30) NOT NULL,
`payroll_id` int(30) NOT NULL,
`name` text NOT NULL,
`amount` float NOT NULL DEFAULT 0,
`taxable` tinyint(1) NOT NULL DEFAULT 0 COMMENT '0=no, 1= yes'

);


CREATE TABLE `deduction_list` (

`deduction_id` int(30) NOT NULL,

`payroll_id` int(30) NOT NULL,

`name` text NOT NULL,

`amount` float NOT NULL DEFAULT 0

);


CREATE TABLE `tax_table_list` (

`tax_id` int(30) NOT NULL,

`payroll_type` int(1) NOT NULL DEFAULT 1 COMMENT '1 = monthly, 2= semi-monthly',

`range_from` float NOT NULL DEFAULT 0,

`range_to` float NOT NULL DEFAULT 0,

`fixed_tax` float NOT NULL DEFAULT 0,

`percentage_over` float NOT NULL DEFAULT 0,

`effective_date` date NOT NULL,

`date_created` timestamp NOT NULL DEFAULT current_timestamp(),

`date_updated` timestamp NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()

);
```

Appendix D: User Interface Screenshots

D.1 Login Interface

The login interface provides secure access to the system with role-based authentication.

D.2 Employee Management Interface

The employee management interface allows administrators to add, edit, and manage employee records with comprehensive information fields.

D.3 Payroll Processing Interface

The payroll processing interface guides users through the multi-step process of calculating and finalizing payroll with interactive forms and real-time calculations.

D.4 Tax Management Interface

The tax management interface enables configuration of tax brackets with effective dates, supporting the dynamic tax calculation system.

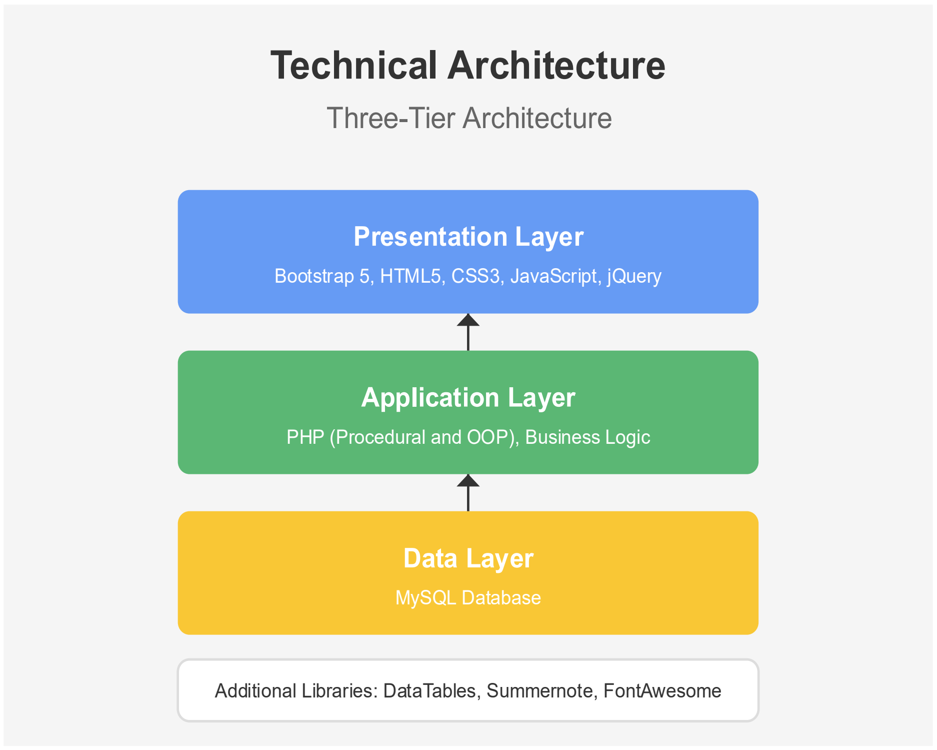

Technical Architecture
Three-Tier Architecture
Presentation Layer
Bootstrap 5, HTML5, CSS3, JavaScript, jQuery
Application Layer
PHP (Procedural and OOP), Business Logic
Data Layer
MySQL Database
Additional Libraries: DataTables, Summernote, FontAwesome

Fig 13 Technical Architecture

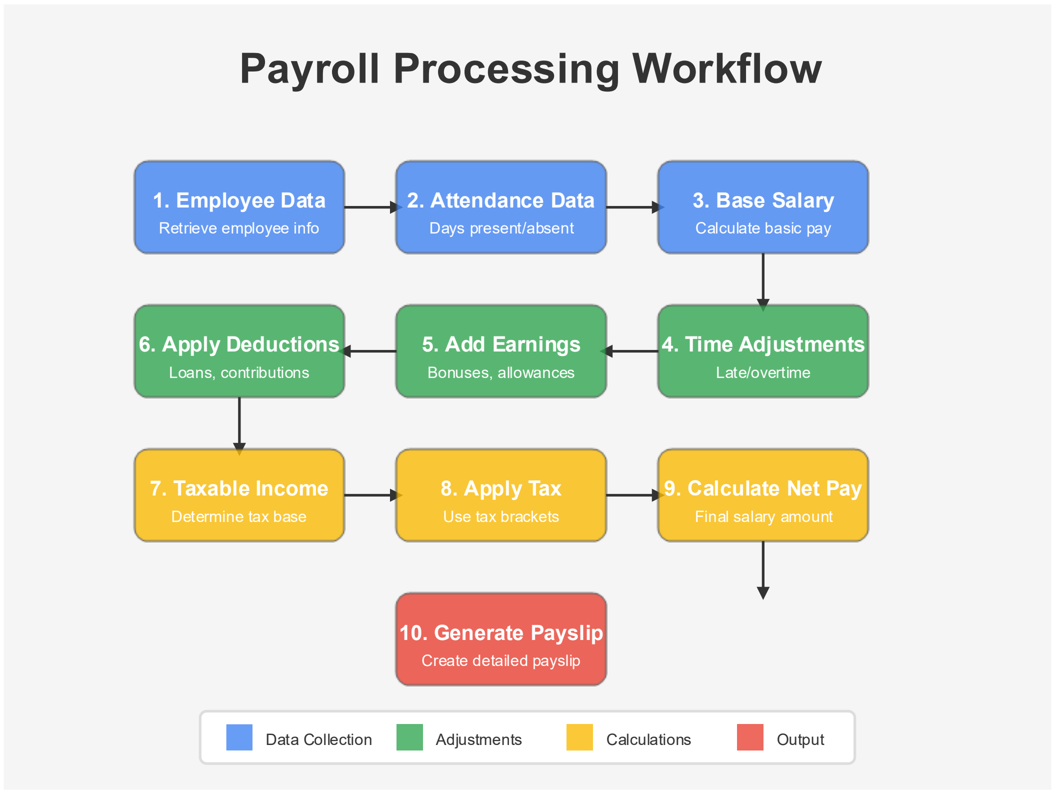


Fig 14 Payroll Processing Workflow