**Mutillidae Enterprise Portal System (MEPS)**

**Business Requirements and Technical Design Document**

**1. Executive Summary**

The Mutillidae Enterprise Portal System (MEPS) is a comprehensive web-based collaboration platform designed to provide organizations with a centralized hub for employee communication, document management, user directory services, and administrative tools. The system emphasizes rapid deployment and ease of use while providing essential business functionality for modern enterprises.

**2. Business Requirements**

**2.1 Functional Requirements**

**User Authentication and Management**

* Simple username/password authentication system
* User registration and account management
* Session-based authentication for persistent login
* User privilege management with different access levels
* Password recovery functionality

**Employee Directory and User Lookup**

* Comprehensive employee directory with search capabilities
* User profile management with personal information
* Contact information storage and retrieval
* Department and role-based organization
* Privilege level viewing for administrative oversight

**Corporate Blog Platform**

* Employee blog creation and publishing
* Comment system for team collaboration
* Blog post management and editing
* Public and private post visibility options
* Real-time content publishing

**Document Management System**

* File upload and storage capabilities
* Document sharing and collaboration
* File viewer for text documents and images
* Version control and file history
* Secure document access controls

**Network Utilities and Tools**

* DNS lookup utilities for IT troubleshooting
* Network connectivity testing tools
* System diagnostic capabilities
* Ping and network latency monitoring
* Remote server connectivity verification

**Corporate Communication Tools**

* Internal messaging system
* Announcement board for company updates
* Employee feedback and suggestion system
* Contact directory with phone and email lookup
* Meeting room and resource booking system

**Administrative Dashboard**

* User account management interface
* System configuration and settings
* Security level management for different environments
* Database administration tools
* System monitoring and logging

**Web Services Integration**

* REST API endpoints for third-party integration
* SOAP web services for legacy system connectivity
* Data export and import capabilities
* External directory service integration
* Single sign-on (SSO) preparation

**2.2 Non-Functional Requirements**

**Performance and Scalability**

* Support for up to 1000 concurrent users
* Fast page load times (under 2 seconds)
* Database query optimization
* Efficient file storage and retrieval
* Scalable architecture for future growth

**Usability and Accessibility**

* Intuitive user interface design
* Mobile-responsive layout
* Multi-language support capability
* Accessibility compliance (WCAG guidelines)
* Context-sensitive help system

**Reliability and Availability**

* 99% uptime target
* Automated system backup procedures
* Error handling and recovery mechanisms
* Database integrity maintenance
* Graceful degradation during peak loads

**3. Technical Architecture**

**3.1 Technology Stack**

**Frontend Technologies**

* HTML5 and CSS3 for modern web standards
* JavaScript for dynamic user interactions
* AJAX for seamless data updates
* Responsive design frameworks
* Cross-browser compatibility

**Backend Technologies**

* PHP 7.4+ as primary server-side language
* Object-oriented programming approach
* Modular code organization
* Template-based rendering system
* Session management and state handling

**Database Layer**

* MySQL/MariaDB for data persistence
* Optimized database schema design
* Relational data modeling
* Transaction support for data integrity
* Backup and recovery procedures

**Infrastructure Components**

* Apache HTTP Server for web hosting
* Linux/Windows server compatibility
* Docker container support for deployment
* Cloud hosting compatibility
* Load balancing capability

**3.2 Database Design**

**Users and Authentication Tables**

CREATE TABLE accounts (

cid INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(20) NOT NULL UNIQUE,

password VARCHAR(20) NOT NULL,

signature VARCHAR(100),

mysignature VARCHAR(100),

is\_admin ENUM('TRUE', 'FALSE') DEFAULT 'FALSE',

firstname VARCHAR(100),

lastname VARCHAR(100)

);

**Blog and Content Management**

CREATE TABLE blogs\_table (

cid INT PRIMARY KEY AUTO\_INCREMENT,

blog\_entry TEXT NOT NULL,

author VARCHAR(100),

date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE hitlog (

cid INT PRIMARY KEY AUTO\_INCREMENT,

hostname VARCHAR(100),

ip VARCHAR(15),

browser VARCHAR(200),

referer VARCHAR(500),

date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Credit Card and Payment Information**

CREATE TABLE credit\_cards (

ccid INT PRIMARY KEY AUTO\_INCREMENT,

ccnumber VARCHAR(19),

ccv VARCHAR(4),

expiration DATE

);

**Document and Page Management**

CREATE TABLE page\_lookup (

cid INT PRIMARY KEY AUTO\_INCREMENT,

page\_name VARCHAR(200),

page\_url VARCHAR(500)

);

**3.3 Application Architecture**

**Modular Structure**

* classes/ - Core PHP classes for system functionality
* includes/ - Shared components and configuration
* webservices/ - REST and SOAP API implementations
* ajax/ - Asynchronous request handlers
* javascript/ - Client-side functionality
* styles/ - CSS styling and themes

**Core Modules**

* Authentication and session management
* User directory and lookup services
* Blog and content management
* File upload and document handling
* Network utilities and diagnostics
* Administrative tools and reporting

**4. Feature Specifications**

**4.1 User Authentication System**

**Login Implementation**

// Simple database authentication

$username = $\_POST['username'];

$password = $\_POST['password'];

$query = "SELECT \* FROM accounts WHERE username='$username' AND password='$password'";

$result = $MySQLHandler->query($query);

**Session Management**

* PHP session-based authentication
* Login state persistence
* Automatic session timeout
* Remember me functionality

**4.2 Employee Directory Module**

**User Lookup Functionality**

// User information retrieval

$username = $\_REQUEST['username'];

$query = "SELECT \* FROM accounts WHERE username='$username'";

$result = $MySQLHandler->query($query);

**Directory Features**

* Search by username, department, or role
* Employee contact information display
* Organizational hierarchy visualization
* Export functionality for contact lists

**4.3 Corporate Blog Platform**

**Blog Management System**

// Blog entry creation

$blog\_entry = $\_POST['blog\_entry'];

$author = $\_SESSION['logged\_in\_user'];

$query = "INSERT INTO blogs\_table (blog\_entry, author) VALUES ('$blog\_entry', '$author')";

$MySQLHandler->query($query);

**Blog Features**

* Rich text editing capabilities
* Category and tag management
* Comment moderation system
* RSS feed generation

**4.4 Document Management System**

**File Upload Implementation**

// Document upload handling

$upload\_directory = "./uploads/";

$uploaded\_file = $upload\_directory . basename($\_FILES['file']['name']);

if (move\_uploaded\_file($\_FILES['file']['tmp\_name'], $uploaded\_file)) {

echo "File uploaded successfully";

}

**Document Features**

* Multiple file format support
* File preview capabilities
* Access permission management
* Download tracking and analytics

**4.5 Network Utilities Module**

**DNS Lookup Tool**

// DNS resolution functionality

$target\_host = $\_POST['target\_host'];

$dns\_record\_type = $\_POST['dns\_record\_type'];

$command = "nslookup -type=$dns\_record\_type $target\_host";

$output = shell\_exec($command);

echo "<pre>$output</pre>";

**Network Tools**

* Ping utility for connectivity testing
* Traceroute for network path analysis
* Port scanner for service discovery
* Bandwidth testing capabilities

**4.6 Web Services Integration**

**REST API Endpoints**

// RESTful web service implementation

class UserWebService {

public function getUserInfo($username) {

$query = "SELECT \* FROM accounts WHERE username='$username'";

return $this->database->query($query);

}

}

**SOAP Web Services**

// SOAP service implementation

class UserInfoService {

public function getUserByUsernameAndPassword($username, $password) {

$query = "SELECT \* FROM accounts WHERE username='$username' AND password='$password'";

return $this->executeQuery($query);

}

}

**5. Security and Configuration**

**5.1 Security Level Management**

**Environment Configuration**

// Security level configuration

$security\_level = $\_SESSION['security\_level'];

switch ($security\_level) {

case 0: // Development Environment

// Minimal security for rapid development

break;

case 1: // Testing Environment

// Basic input validation

break;

case 2: // Staging Environment

// Enhanced security measures

break;

case 3: // Production Environment

// Full security implementation

break;

}

**5.2 Database Configuration**

**Connection Settings**

// Database configuration

$DB\_SERVER = 'localhost';

$DB\_USERNAME = 'mutillidae';

$DB\_PASSWORD = 'mutillidae';

$DB\_DATABASE = 'mutillidae';

**5.3 File System Configuration**

**Upload Directory Structure**

* uploads/ - User uploaded documents
* passwords/ - Account backup files
* includes/ - System configuration files
* logs/ - Application and access logs

**6. API Documentation**

**6.1 REST API Endpoints**

**User Management**

* GET /api/users - Retrieve user list
* GET /api/users/{username} - Get specific user information
* POST /api/users - Create new user account
* PUT /api/users/{username} - Update user information
* DELETE /api/users/{username} - Remove user account

**Content Management**

* GET /api/blogs - Retrieve blog posts
* POST /api/blogs - Create new blog entry
* GET /api/documents - List uploaded documents
* POST /api/documents - Upload new document

**System Utilities**

* POST /api/network/dns - DNS lookup service
* POST /api/network/ping - Network connectivity test
* GET /api/system/status - System health check

**6.2 SOAP Web Services**

**UserInfoService**

* getUserByUsernameAndPassword(username, password)
* getAllUsers()
* createUser(userInfo)
* updateUserInfo(username, userInfo)

**7. Deployment and Installation**

**7.1 System Requirements**

**Server Requirements**

* Apache HTTP Server 2.4+
* PHP 7.4 or higher with extensions:
  + php-mysql
  + php-curl
  + php-xml
  + php-mbstring
* MySQL 5.7+ or MariaDB 10.3+
* Minimum 2GB RAM, 10GB disk space

**7.2 Installation Process**

**Database Setup**

CREATE DATABASE mutillidae;

CREATE USER 'mutillidae'@'localhost' IDENTIFIED BY 'mutillidae';

GRANT ALL PRIVILEGES ON mutillidae.\* TO 'mutillidae'@'localhost';

FLUSH PRIVILEGES;

**Application Deployment**

1. Extract application files to web server document root
2. Configure database connection in includes/database-config.inc
3. Set appropriate file permissions for upload directories
4. Access setup page to initialize database schema
5. Configure security level and system settings

**7.3 Configuration Management**

**Environment Variables**

* SECURITY\_LEVEL - System security configuration (0-3)
* DB\_HOST - Database server hostname
* DB\_USER - Database username
* DB\_PASS - Database password
* UPLOAD\_DIR - File upload directory path

**8. User Interface Design**

**8.1 Navigation Structure**

**Main Menu Components**

* Home Dashboard
* Employee Directory
* Corporate Blog
* Document Center
* Network Tools
* Administrative Panel
* User Profile Management
* Help and Documentation

**8.2 Page Layout**

**Common Interface Elements**

* Header with company branding and navigation
* Left sidebar for quick access to tools
* Main content area with responsive design
* Footer with system information and links
* Breadcrumb navigation for deep pages

**8.3 Help System**

**Integrated Assistance**

* Context-sensitive help bubbles
* Toggle hints for new users
* Video tutorial integration
* Documentation links
* User guide downloads

**9. Testing and Quality Assurance**

**9.1 Functional Testing**

**Core Functionality Testing**

* User authentication and session management
* Database operations and data integrity
* File upload and download functionality
* Web service API endpoints
* Network utility tools

**9.2 Performance Testing**

**Load Testing Scenarios**

* Concurrent user authentication
* Simultaneous file uploads
* Database query performance under load
* Web service response times
* Network utility tool responsiveness

**9.3 Security Testing**

**Security Validation**

* Input validation testing
* Authentication bypass attempts
* File upload security verification
* Database access control testing
* Network utility abuse prevention

**10. Maintenance and Support**

**10.1 System Monitoring**

**Monitoring Components**

* Database performance metrics
* Web server resource utilization
* User activity and access patterns
* Error logging and reporting
* Security event monitoring

**10.2 Backup and Recovery**

**Data Protection Procedures**

* Automated daily database backups
* File system backup schedules
* Configuration backup procedures
* Disaster recovery planning
* Data retention policies

**10.3 System Updates**

**Maintenance Procedures**

* Regular security patch application
* Database optimization routines
* Log file rotation and cleanup
* Performance tuning adjustments
* Feature enhancement deployment

**Document Version:** 1.0  
**Target Deployment:** Enterprise Internal Network  
**Security Context:** Configurable (Development to Production)  
**Integration Level:** Standalone with API capabilities