ZUKBITS WORKFLOWS SYSTEM

Master Specification, Requirements, and Architecture Document

Version 1.0: Internal Use Only

Prepared for: ZukBits Online – "Make It Happen"

Prepared by: System Admin(Nzioka Emmanuel)

Document Type: Unified SRS + SDS + Architectural & Operational Guide

Classification: Internal Confidential

Monday, December 8, 2025e

General Introduction

The ZukBits Workflows System is an internal enterprise platform designed to organize, track, document, and enhance all operational workflows across ZukBits Online. The system addresses fundamental challenges in the organization’s project execution and staff performance processes, including fragmented project tracking, inconsistent documentation practices, lack of centralized approvals, and limited visibility into weekly staff activities.

As the company continues to expand its client base, internal digital platforms, and innovation-driven solutions, maintaining operational structure becomes critical. The ZukBits Workflows System acts as the single source of truth for projects, performance, documentation, communication, and decision flows, ensuring that the organization operates with clarity, accountability, efficiency, and long-term sustainability.

Built with simplicity yet engineered for growth, the platform incorporates role-based access, secure data handling, project lifecycle automation, and comprehensive dashboards tailored for each stakeholder group—including Developers, Marketers, System Administrators, Directors, and the Super Admin. Each module, from project creation to approval workflows, weekly schedules, reporting, and technical documentation, is structured to support seamless collaboration and effective oversight.

The goal of the ZukBits Workflows System is not only to streamline daily operations but also to preserve organizational knowledge, enhance staff planning and reporting discipline, and provide executive management with real-time visibility into all ongoing and completed activities across the company.

This document serves as the end-to-end technical and operational blueprint for the ZukBits Workflows System. It combines:

- Software Requirements Specification (SRS)
- Software Design Specification (SDS)
- System Architecture & Module Definitions

- Security, Testing, Deployment, and Maintenance Guidelines
- Role-based dashboards and workflow descriptions

This single consolidated master document is intended to guide:

- System designers
- Backend & frontend developers
- System administrators
- Directors and decision-makers
- Future maintainers
- Auditors and reviewers

Through detailed descriptions, requirements, diagrams, and architectural breakdowns, the document ensures that every aspect of the system; from UX behavior to database structure and security protocols; is explicitly defined and aligned with ZukBits' operational goals.

The system represents the organizational culture of ZukBits Online: innovative, organized, transparent, and digitally enabled. It supports the company's mission to deliver exceptional digital solutions while maintaining strong, scalable, and reliable internal operations.

Table of Contents

Master Specification, Requirements, and Architecture Document

SECTION 1: SRS (SOFTWARE REQUIREMENTS SPECIFICATION)

1.1 Purpose

This document defines the functional and non-functional requirements for the ZukBits Internal Project & Performance Management System (“the System”). It serves as the single source of truth for business stakeholders, developers, system admins, and future maintainers.

The System will be used internally by ZukBits Online – “The Home of Innovations” to manage:

- Projects and technical work
- Documentation and knowledge
- Weekly schedules and targets
- Weekly reports and performance
- Approvals and decision flows
- Communication around work
- Dashboards and analytics for management

1.2 Scope

The System will:

- Centralize all projects, staff performance data, and documentation.
- Implement role-based access for:
  - Super Admin
  - Director
  - System Admin

    - Developer
    - Marketer
- Provide dashboards tailored to each role.
- Support weekly planning and reporting.
- Store and protect technical details (API keys, credentials, configs).

The system will initially be developed with:

- Backend: PHP
- Frontend: HTML + Bootstrap CSS + vanilla JavaScript
- Database: PostgreSQL (Supabase in dev, AWS RDS Postgres or MySQL in production)
- Dev hosting: Vercel (or similar)
- Prod hosting: AWS (separate deployment)

1.3 Stakeholders

- Super Admin – Responsible for system-level configuration and account lifecycle management.
- Director – Executive management who needs visibility into all operations and performance.
- System Admin – Handles platform configuration, project assignment workflows, and operational setup.
- Developers – Execute technical work on projects, maintain documentation, and report progress.
- Marketers – Execute marketing work and report weekly activities and results.

1.4 System Overview

The System acts as a central hub for:

- Project lifecycle tracking (creation → work → completion → approval → archive)
- Technical documentation and secure credential storage
- Weekly schedules and targets
- Weekly reports and performance data
- Notifications and communication
- Management dashboards and analytics

2.0 User Roles & Permissions (Conceptual)

2.1 Super Admin

- Full visibility across all modules.
- Create, update, deactivate, delete user accounts.
- Assign roles (Super Admin, Director, System Admin, Developer, Marketer).
- Configure global system settings (company profile, environment toggles, etc.).
- Override access if necessary (e.g., temporarily read-only access to any record).

2.2 Director

- View all projects and their statuses.
- View all staff weekly schedules and reports.
- Approve or reject project completion.
- Approve or comment on critical decisions.
- View performance dashboards for team and individuals.

2.3 System Admin

- Configure non-critical system settings (labels, categories, project types).
- Manage departments, project categories (internal, client, feature, etc.).

- Assist in project assignment to developers and marketers.
- Manage documentation categories and metadata.
- Ensure backups and health checks (operationally, not infrastructure-level).

2.4 Developer

- View and manage projects assigned to them.
- Update project milestones, tasks, timelines, and blockers.
- Upload and update technical documentation (architecture, credentials, configs).
- Submit project as "Completed – Pending Approval".
- Submit weekly schedules (planned work).
- Submit weekly reports (achieved vs planned).

2.5 Marketer

- View marketing-related projects/campaigns assigned to them.
- Submit weekly schedules (planned campaigns, content, tasks).
- Submit weekly reports (results, reach, conversions, visibility).
- Attach marketing assets, links, and references.

3.0 Functional Requirements

3.1 Authentication & User Management

- FR-1: The system shall allow users to log in using email + password credentials.
- FR-2: The system shall support password reset via email.
- FR-3: The system shall allow Super Admin to create, edit, deactivate, and delete user accounts.
- FR-4: The system shall allow Super Admin to assign or change roles.
- FR-5: The system shall enforce role-based access control (RBAC).

3.2 Project Management Module

FR-10: Authorized users (Developers/System Admin) shall be able to create projects with the following fields:

I. Project name
II. Project code (auto or manual)
III. Client / internal unit
IV. Description & objectives
V. Start date
VI. Target end date
VII. Assigned developer(s)
VIII. Assigned marketer(s) (optional)
IX. Priority (High/Medium/Low)
X. Category (Internal system, client project, feature, marketing campaign)

- FR-11: The system shall maintain project status: Draft, Ongoing, Completed – Pending Approval, Approved, Archived.
- FR-12: Developers shall be able to update project milestones, tasks and progress logs.
- FR-13: The system shall allow attaching files (e.g., briefs, contracts, screenshots).
- FR-14: System shall support a progress log: timestamp, user, comment, optional file.
- FR-15: Developer shall mark project as “Completed – Pending Approval” when done.
- FR-16: Director and Super Admin shall be able to review and either Approve or Reject the completion.
- FR-17: Upon approval, the project shall move to “Done / Archived”.

3.3 Technical Documentation & Knowledge Hub

- FR-20: Developers shall be able to store documentation per project, including:
    I. System architecture notes
    II. API endpoints & documentation
    III. Third-party integrations
    IV. Configuration files (non-secret)
    V. Environment details: dev/stage/prod
    VI. User accounts & credentials (encrypted and access-controlled)
- FR-21: The system shall store credentials in encrypted form and only allow access to authorized roles.
- FR-22: System Admin and Super Admin shall be able to define categories/tags for documentation.
- FR-23: The system shall allow searching documentation by project, tag, keyword.

3.4 Weekly Schedules & Targets

- FR-30: All staff (Developers and Marketers, at minimum) shall be able to submit a weekly schedule:
    I. Week start date
    II. Planned tasks
    III. Time allocation per task (optional)
    IV. Weekly targets (quantitative or qualitative)
- FR-31: The system shall allow editing a schedule only until a locking time (e.g., end of Monday).
- FR-32: Directors and System Admins shall be able to view all weekly schedules.

3.5 Weekly Reports & Performance Tracking

- FR-40: At the end of each week, staff shall submit a weekly report:
    I. Achieved vs planned tasks
    II. What was not completed & reasons
    III. Challenges and blockers
    IV. Support needed
- FR-41: The system shall auto-link the weekly report to that week's schedule.
- FR-42: The system shall generate summaries per user and per team.
- FR-43: Director shall see performance trends across time for each staff member.

3.6 Approvals

- FR-50: The system shall support approval workflows for:
    I. Project completion
    II. Key decisions / milestones (optional)
- FR-51: Approvers shall be Director + Super Admin by default.
- FR-52: Approvers shall be able to leave comments on approval/ rejection.
- FR-53: System shall record approval history (who approved, when, what decision).

3.7 Communication & Notifications

- FR-60: The system shall show in-app notifications for:
    I. New project assignments
    II. Project status changes
    III. Approval requests and decisions
    IV. Weekly schedule/report reminders
- FR-61: The system shall optionally send email notifications for the same events.

- FR-62: Each project shall have a simple comment thread for clarifications.
- FR-63: Comments shall support @mention to notify specific users.

3.8 Dashboards & Reporting

- FR-70: The system shall provide role-based dashboards:
  I. Super Admin dashboard
  II. Director dashboard
  III. System Admin dashboard
  IV. Developer dashboard
  V. Marketer dashboard
- FR-71: Director dashboard shall show:
  I. Ongoing vs completed projects
  II. Overdue projects
  III. Weekly report completion rates
  IV. Performance summaries by staff
- FR-72: System Admin dashboard shall show:
  I. System health info (basic)
  II. Project assignment overview
  III. Documentation coverage per project
- FR-73: Developer dashboard shall show:
  I. Assigned projects
  II. Upcoming deadlines
  III. Weekly schedule status

IV. Recent feedback

- FR-74: Marketer dashboard shall show:

I. Assigned campaigns/projects

II. Weekly schedule and reports summary

III. Simple metrics (if tracked manually in reports)

- FR-75: Reports shall be exportable as PDF / CSV.

3.9 Security & Data Protection

- FR-80: Access to credentials/API keys shall be limited to authorized roles (e.g., Super Admin, System Admin, specific Developers).
- FR-81: The system shall encrypt sensitive fields at rest (e.g., credentials).
- FR-82: The system shall log access to sensitive data.
- FR-83: The system shall have regular backups (daily minimum).
- FR-84: Session management shall include expiration and protection against common attacks (CSRF, XSS, etc.).

4.0 Non-Functional Requirements

- **NFR-1: Performance**
    - The system should support at least 50 concurrent active users with acceptable response times (< 2 seconds for standard pages).
- **NFR-2: Availability**
    - Target uptime 99% for internal usage (excluding planned maintenance).
- **NFR-3: Scalability**
    - Must be scalable to support additional roles and departments without fundamental redesign.
- **NFR-4: Usability**

- Simple, clear Bootstrap-based interfaces.
- Minimal clicks to perform common tasks.

- **NFR-5: Security**
  - Follow best practices for password storage (hash+salt).
  - Encrypted database fields for sensitive info.
  - Access control at backend level (not just UI).
- **NFR-6: Maintainability**
  - Clean PHP structure (MVC-like separation).
  - Configurable environment variables.
- **NFR-7: Portability**
  - Should run in dev (Supabase + Vercel) and production (AWS + RDS) with minimal configuration differences.

5.0 Key Use Cases (Summary)

- UC-1: Super Admin creates a new user and assigns a role.
- UC-2: Developer creates a new project and assigns themselves or others.
- UC-3: Developer updates milestones and marks project as “Completed – Pending Approval.”
- UC-4: Director reviews and approves project completion.
- UC-5: Staff submits weekly schedule and weekly report.
- UC-6: Director views performance dashboard for a selected period.
- UC-7: Developer uploads technical documentation and credentials.
- UC-8: System Admin configures categories, priorities, and project types.

That closes the SRS section.

SECTION 2: SDS (SOFTWARE DESIGN SPECIFICATION)

This section defines how we will implement the system described in the SRS.

6.0 Overall Design Approach

- **Architecture style:**
  - Server-rendered PHP application with clear separation of concerns (MVC-like pattern).
- **Frontend:**
  - HTML templates with Bootstrap for layout and components.
  - Vanilla JavaScript for interactivity (modals, AJAX for a few actions like notifications).
- **Backend:**
  - PHP acting as HTTP handlers/controllers for each module.
- **Database:**
  - PostgreSQL (recommended, to match Supabase dev and AWS RDS Postgres in prod).
- **Deployment:**
  - Dev: Supabase + Vercel (via PHP runtime or separate backend hosting).
  - Prod: AWS (e.g., EC2/Lightsail/Fargate) + RDS.

7.0 High-level Architecture

7.1 Layers

1. **Presentation Layer (Frontend)**
   - PHP/HTML templates, Bootstrap CSS, JS scripts.
   - Provides forms, dashboards, modals, tables, search/filter UI.

2. **Application Layer (Business Logic)**
    - PHP classes or functions implementing use cases: project creation, approval, report submission, etc.
    - Enforces role-based rules.
3. **Data Access Layer**
    - PHP repository/DAO classes to interact with PostgreSQL/MySQL using PDO.
    - Prepared statements for security.
4. **Database Layer**
    - PostgreSQL or MySQL schemas.
    - Tables for users, roles, permissions, projects, weekly_schedules, weekly_reports, documents, credentials, notifications, logs, etc.

8.0 Core Components

8.1 Authentication Component

- Handles login, logout, password reset.
- Session management, CSRF protection tokens on forms.
- Verifies role and user status (active/inactive).

8.2 RBAC (Role-Based Access Control) Component

- Maintains a mapping of *role to permissions*.
- **Example permission groups:**
    - manage_users
    - view_all_projects
    - edit_assigned_projects

  - view_performance_dashboard
  - manage_system_settings
- **Middleware-like logic on each route/controller:**
  - Check if user is authenticated.
  - Check if user's role has required permission.
  - If not, redirect to "Access Denied" page.

8.3 Project Module Component

- **Controllers:**
  - ProjectController – handles listing, creating, editing, viewing, updating, status transitions.
- **Views:**
  - projects_list.php, project_detail.php, project_form.php, etc.
- **Data Access:**
  - ProjectRepository with methods: getAll, getById, create, update, delete, getByUser, getByStatus, etc.

8.4 Documentation & Credential Component

- DocumentationController – for managing text docs, files, and metadata.
- CredentialsController – for sensitive data.
- Encryption utilities in PHP for encrypting/decrypting specific fields:
  - e.g., encrypt($plaintext) & decrypt($ciphertext) using AES-256-GCM with a key stored in environment variable.
- Access checks to ensure only authorized roles can view credentials.

8.5 Weekly Schedule & Report Components

- ScheduleController – create, view, edit weekly schedules.
- ReportController – create, view weekly reports.
- Link weekly_reports.week_id to weekly_schedules.id.

8.6 Notification Component

- NotificationController – list notifications for logged-in user.
- NotificationService – create notifications on specific events:
    - New assignment
    - Approval requests
    - Due date approaching
- Uses database table notifications with fields:
    - id, user_id, type, message, link, is_read, created_at.

8.7 Dashboard Component

- Each role leads to a different dashboard page:
    - /dashboard/superadmin
    - /dashboard/director
    - /dashboard/systemadmin
    - /dashboard/developer
    - /dashboard/marketer
- Each dashboard pulls widgets using dedicated services/repositories.

9.0 Data Model (Key Tables – High-level)

Note: Exact SQL not shown here, but the structure is clearly defined.

9.1 Users & Roles

- **users**
    - id (PK)
    - name
    - email
    - password_hash
    - role_id (FK → roles.id)
    - is_active (bool)
    - created_at
    - updated_at
- **roles**
    - id (PK)
    - name (Super Admin, Director, System Admin, Developer, Marketer)
    - created_at

Optional: role_permissions to allow future fine-grained control.

9.2 Projects

- **projects**
    - id (PK)
    - name
    - code
    - client_name
    - description

  - start_date
  - target_end_date
  - status (enum: draft, ongoing, pending_approval, approved, archived)
  - priority (high/medium/low)
  - category (internal, client, feature, campaign, etc.)
  - created_by (FK → users.id)
  - created_at
  - updated_at

- **project_assignments**
  - id (PK)
  - project_id (FK → projects.id)
  - user_id (FK → users.id)
  - role_type (developer, marketer)
- **project_progress_logs**
  - id (PK)
  - project_id (FK)
  - user_id (FK)
  - comment
  - status_snapshot (optional)
  - created_at

9.3 Documentation & Credentials

- **project_documents**

  - id (PK)
  - project_id (FK)
  - title
  - type (architecture, integration, notes, etc.)
  - body (text)
  - file_path (optional)
  - tags (text array or CSV)
  - created_by
  - created_at

- **project_credentials**
  - id (PK)
  - project_id (FK)
  - label (e.g., “Production DB”, “Stripe API Key”)
  - encrypted_value
  - description
  - allowed_roles (optional JSON or text)
  - created_by
  - created_at

9.4 Weekly Schedules & Reports

- **weekly_schedules**
  - id (PK)
  - user_id (FK)

  - week_start_date
  - summary_plan
  - created_at
- **weekly_schedule_items**
  - id (PK)
  - schedule_id (FK)
  - description
  - estimated_hours (optional)
  - related_project_id (FK projects.id, nullable)
- **weekly_reports**
  - id (PK)
  - schedule_id (FK → weekly_schedules.id)
  - user_id (FK)
  - overall_summary
  - challenges
  - support_needed
  - submitted_at
- **weekly_report_items**
  - id (PK)
  - report_id (FK)
  - schedule_item_id (FK)
  - status (completed, partially_completed, not_completed)

- comment

9.5 Approvals

- **approvals**
    - id (PK)
    - approval_type (project_completion, milestone, etc.)
    - target_id (e.g., project_id)
    - requested_by (FK users.id)
    - status (pending, approved, rejected)
    - created_at
- **approval_decisions**
    - id (PK)
    - approval_id (FK → approvals.id)
    - approver_id (FK users.id)
    - decision (approved, rejected)
    - comment
    - decided_at

9.6 Notifications

- notifications
    - id (PK)
    - user_id (FK)
    - type
    - message

    - link
    - is_read (bool)
    - created_at

10.0 Key Design Decisions

- Use PostgreSQL to align with Supabase in dev and AWS RDS in prod.
- Use server-rendered PHP to reduce complexity and improve reliability.
- Use Bootstrap to keep UI consistent and fast to build.
- Store credentials encrypted with environment-based keys.
- Implement RBAC in backend to avoid relying on frontend checks.

That closes the SDS core. The next part goes deeper into modules & dashboards UX, plus deployment, security, testing, and maintenance in a more operational view.

SECTION 3: MODULES & DASHBOARDS

11.0 Modules (Full Breakdown)

11.1 Authentication & User Management Module

- **Main screens:**
  - Login page
  - Forgot password page
  - User management (Super Admin only)
- **Super Admin actions:**
  - Create new user (name, email, role, active status)
  - Deactivate/activate user
  - Change user role
- **Validation & UX:**
  - Inline validation messages
  - Clear error messages for login failure ("Invalid email or password")
  - Status badge (Active / Inactive) in user list.

11.2 Project Management Module

- **Screens:**
  - Project list page (filters by status, category, priority, owner).
  - Project detail page (with tabs: Overview, Milestones, Documents, Credentials, Comments, History).
  - Project creation/edit form.
- **Fields:**

    - Name, Code, Client, Description, Objectives, Start/End date, Category, Priority, Assigned users, Status.

- **Logic:**
    - Only Super Admin, Director, and System Admin can assign users to a project (configurable).
    - Developers can create projects but only assign themselves (optional rule).
    - Status transitions:
        - Draft->Ongoing → Pending Approval → Approved → Archived
    - Business rule: Only Director or Super Admin can approve a Pending Approval project.

11.3 Documentation & Knowledge Hub Module

- **Screens:**
    - Project documentation list (per project).
    - Documentation detail view (title, content, attachments).
    - Documentation editor (create/edit).
    - Credential list (protected page).
    - Credential detail (with security gate such as “Click to reveal” + log event).
- **UX:**
    - Use tabs or side navigation inside project to access Documentation.
    - Search bar for documentation (by title/content/tags).
- **Security:**
    - Must check user’s role before showing credential list or allowing decryption.

11.4 Weekly Schedules & Targets Module

- **Screens:**
  - “My Weekly Schedule” list (by week).
  - Weekly schedule form (add tasks, targets).
  - Manager (Director/System Admin) view of all schedules by user and week.
- **UX:**
  - For each week: show a card with summary targets and “View details”.
  - In details: show list of tasks with estimated hours and associated projects.

11.5 Weekly Reports & Performance Module

- **Screens:**
  - “My Weekly Reports” list.
  - Report form (linked to a specific schedule).
  - Manager’s view of staff reports.
  - Performance summary dashboard (per staff).
- **Logic:**
  - Reports must always be linked to an existing schedule (no orphan reports).
  - Reports can only be submitted once per week per user.
  - Manager can comment on a report (optional).

11.6 Approvals Module

- **Screens:**
  - Approvals inbox (for Director & Super Admin).
  - Approval detail page (show project summary, documentation links, logs).

  - Actions: Approve / Reject with comment.

- **UX:**
  - Show badges like “Pending Approval”, “Approved”, “Rejected”.
  - Show timeline of approval decisions.

11.7 Communication & Notification Module

- **Screens:**
  - Notifications dropdown in navbar.
  - Full notifications page with filters (Unread, All, By type).
  - Project comments section as part of Project detail.
- **UX:**
  - Notification bell icon with unread counter.
  - Clicking a notification opens the relevant page (project, approval, schedule, etc.).

11.8 Dashboards Module (Per Role)

11.8.1 Super Admin Dashboard

- **Widgets:**
  - Total users by role (Super Admin / Director / System Admin / Developer / Marketer).
  - Active vs inactive users.
  - Number of ongoing vs completed projects.
  - System-level alerts (e.g., “X users haven’t submitted weekly report”).
- **Tables:**
  - Recent projects created.

  - Users created/updated recently.

11.8.2 Director Dashboard

- Widgets:
  - Ongoing vs completed vs overdue projects.
  - Weekly report submission rate (e.g., 80% of staff submitted last week).
  - Staff performance index (based on completion rates and approvals).
- **Tables:**
  - "Projects at Risk" – projects with overdue target end date or flagged blockers.
  - "This Week's Reports" – list of all submitted weekly reports.
- **Filters:**
  - By department (if added later), by role, by project category.

11.8.3 System Admin Dashboard

- **Widgets:**
  - Number of projects missing documentation.
  - Number of projects with stored credentials.
  - Backups/health indicators (if integrated later).
- **Tables:**
  - "Projects without assigned developers."
  - "Projects without recent updates (e.g., >14 days)."

11.8.4 Developer Dashboard

- **Widgets:**
  - Count of assigned projects (by status).

  - Next 7 days deadlines.
  - Weekly schedule status (submitted / not submitted).
- **Tables:**
  - "My Ongoing Projects."
  - "My Tasks this Week."
  - "Recent Feedback / Comments on my projects."

11.8.5 Marketer Dashboard

- **Widgets:**
  - Number of active campaigns/projects.
  - Weekly schedule submission status.
  - Quick stat placeholders (e.g., manually input campaign results in reports).
- **Tables:**
  - "My Active Projects/Campaigns."
  - "My Weekly Reports Summary."

12.0 Navigation & UX Patterns

- **Top Navbar:**
  - Brand logo/name
  - Links to: Dashboard, Projects, Weekly, Reports, Documentation (if authorized), Admin (if authorized)
  - Notification bell
  - User profile menu (Profile, Logout)
- **Sidebar (optional):**
  - Role-based navigation items.

- **UI Consistency:**
    - Use Bootstrap for consistent buttons, forms, alerts.
    - Use icons sparingly, focus on clarity.

SECTION 4: DEPLOYMENT, SECURITY, TESTING, MAINTENANCE & FINAL SUMMARY

13.0 Environments & Deployment

13.1 Development Environment

- Backend: PHP running locally (XAMPP, Laravel Valet, or Docker).
- Database: Supabase (hosted Postgres).
- Hosting: Vercel for frontend or combined PHP app (or another dev-friendly host).
- Config: .env file containing Supabase DB connection details, encryption key, mail creds.

13.2 Production Environment

- Hosting: AWS (EC2/Lightsail or ECS/Fargate).
- Database: AWS RDS (PostgreSQL recommended).
- Config:
  - Environment variables set via AWS SSM or similar.
  - Different encryption key than dev.
  - Logging to CloudWatch or another centralized system.

13.3 Migration from Dev to Prod

- Use PostgreSQL in both Supabase and RDS to minimize schema differences.
- Use SQL migrations files (e.g., Flyway or a custom PHP migration script).
- Steps:
  1. Export schema and data from Supabase (for initial setup).
  2. Import into AWS RDS.
  3. Update .env to point to AWS database.

4. Run a smoke test.

14.0 Configuration Management

- All secrets (DB password, encryption keys, SMTP credentials) stored in environment variables.
- Never commit secrets into git.
- Have separate config per environment:
    - APP_ENV=local|dev|prod
    - DB_HOST, DB_NAME, DB_USER, DB_PASS
    - ENCRYPTION_KEY
    - MAIL_HOST, MAIL_USER, MAIL_PASS

15.0 Security Design & Operations

- **Authentication:**
    - Passwords hashed using bcrypt/argon2.
    - Sessions managed securely with HTTP-only cookies.
- **Authorization:**
    - Enforce RBAC in every controller.
    - Deny by default, allow only if permission is granted.
- **Data Protection:**
    - Encrypt credentials and sensitive fields.
    - Log every access to credentials (who, when, which project).
- **Input Validation & Sanitization:**
    - Validate all forms server-side.
    - Sanitize user input before output (escape HTML).

- **Transport Security:**
  - Use HTTPS in production (TLS certificate).
- **Backups & Recovery:**
  - Daily automated DB backups.
  - Monthly backup verification test (restore to dev/stage).

16.0 Testing Strategy

16.1 Types of Tests

- Unit tests for core business logic (PHP).
- Integration tests for database interactions.
- End-to-end tests for core flows:
  - Login
  - Project creation & approval
  - Weekly schedule & report
  - Documentation and credentials access

16.2 UAT (User Acceptance Testing)

- Key internal users (Director + selected Developer + System Admin) will test:
  - Usability
  - Clarity of dashboards
  - Correctness of flows

16.3 Performance Testing

- Test listing pages with sample data (e.g., 1000 projects) to ensure acceptable performance.
- Optimize queries with indexes where needed.

17.0 Maintenance & Evolution

- **Codebase:**
    - Use git for version control.
    - Establish branching model (main, develop, feature branches).
- **Change Management:**
    - All changes should be documented in a CHANGELOG.
    - Database schema changes through migration scripts.
- **Feature Roadmap (future):**
    - API endpoints for integration with other tools.
    - More advanced analytics (graphs, charts).
    - SSO or OAuth integration.
    - Mobile-optimized views or PWA.

18.0 Final Summary

This master document defines:

- **The business problem**: scattered projects, lack of visibility, no standardized approvals, and weak performance tracking.
- **The solution**: a centralized internal workflows platform that handles projects, documentation, weekly schedules & reports, approvals, communication, and dashboards, with clear roles and security.
- The roles:
    - Super Admin – full control of accounts and system.
    - Director – executive visibility and approvals.
    - System Admin – platform configuration and operational management.

  - Developers – technical execution and documentation.
  - Marketers – campaign execution and reporting.
- **The modules:**
  - Authentication & RBAC
  - Project management
  - Documentation & credentials
  - Weekly schedules & reports
  - Approvals
  - Notifications & communication
  - Role-based dashboards
- The design & architecture: PHP + Bootstrap + vanilla JS, PostgreSQL, RBAC, encrypted credentials, multi-environment setup (Supabase+Vercel in dev, AWS+RDS in prod).
- The operational aspects: deployment, security, testing, backups, and future improvements.