MIGRATEFORGE

Assessment

Sample Paid Assessment Deliverable — WA Blast Pro (26,436 LOC Plain PHP → Java 21)

This is a benchmark application, not a client project. It is published as an example of the deliverable.

26,436 PHP lines · medium · 13 November 2026

Prepared by MigrateForge

What this assessment is

- What was analysed: the PHP source graph, schema, risks and configuration.
- What was recorded: HTTP journeys and outgoing third-party calls when a capture was supplied.
- What is proposed: a Java 21 + Spring Boot target shape and a vertical-slice wave plan.
- What is still open: questions the code cannot answer, listed in the open questions register.

What it is not

- Not a migration — no Java was generated for this pack.
- Not a quote for work outside the stated scope.
- Requirements are limited to facts present in the codebase; uncited statements become open questions.

Contents

1. Executive summary 4
2. Business requirements 5
3. Data model 8
4. Technical assessment 10
5. Target architecture 12
6. Migration plan 15
7. Risk register 17
8. Test and parity strategy 18
9. Database plan 19
10. Open questions 20
11. Stakeholder interview guide 22
12. Open questions register 24
13. What happens next 29

1. Executive summary

WA Blast Pro is a plain-PHP, multi-tenant WhatsApp marketing and CRM SaaS. It has **26,436 PHP LOC** and is sized **MEDIUM**. This assessment covers its migration to **Java 21**.

Functional scope (12 planned modules):

- common, users
- inbox, ai
- leads, campaigns, flows
- appointments, rewards
- billing, saas, mobileapi

Key characteristics

- **Tenancy:** database-per-tenant. A master DB holds `tenants`, `plans`, `invoices` and `usage_logs`. Each tenant gets its own DB with encrypted credentials.
- **Integrations:**
  – WhatsApp Cloud API (`WhatsAppAPI`, complexity 189)
  – Razorpay (subscriptions, orders, signature checks)
  – AI chatbot engine (`AIEngine`, complexity 90)
- **Background processing:** 8 cron scripts handle:
  – campaigns
  – flows
  – appointment reminders
  – birthday wishes
  – tier checks
  – billing
- **Mobile REST API:** v1 and v2, with JWT, refresh tokens and idempotent sends.
- **Behavioural baseline:** 660 recorded fixtures across ~66 journeys. Journey 1 alone has 289 fixtures.

Complexity hotspots

Unit	Complexity	LOC
inbox/index.php	192	1028
`WhatsAppAPI`	189	346
processEvent (webhook)	134	277
appointments/index.php	120	419
inbox/logs.php	113	348
handleBookingInbound	81	155

Top risks

- **64 dynamic includes:** concentrated in cron scripts, the mobile API bootstrap and the rewards includes.
- **3 sleep-in-loop sites:** campaign senders and the auto-reply path.

- **Diagnostic scripts:** `diag.php` and `fix.php` are exposed as routes.

Requirements

- The migration must preserve all 12 functional modules. Each module maps to a Java package under com.wablastpro.*.
  Source: plan.modules:common, plan.modules:users, plan.modules:inbox, plan.modules:ai, plan.modules:leads, plan.modules:campaigns, plan.modules:flows, plan.modules:appointments, plan.modules:rewards, plan.modules:billing, plan.modules:saas, plan.modules:mobileapi
- Behavioural parity must be verified against the 660 recorded capture fixtures. This includes the 289-fixture journey 1.
  Source: capture:bb058f1f-c343-445f-aa22-a1e9257fb203, capture.journey:1
- WhatsApp messaging must be migrated with equivalent semantics, covering both outbound API calls and inbound webhook event processing.
  Source: class:WhatsAppAPI@config/whatsapp.php:5, script:includes/webhook_processor.php@includes/webhook_processor.php:1, function:processEvent@includes/webhook_processor.php:8
- The Razorpay billing integration must be migrated, including webhook and payment signature verification.
  Source: function:razorpayVerifyWebhookSignature@includes/razorpay.php:79, function:razorpayVerifyPaymentSignature@includes/razorpay.php:85, route:ANY /billing/webhook.php@billing/webhook.php:1
- All 64 dynamic includes must be resolved into static dependencies. All 3 sleep-in-loop sites must be eliminated during migration.
  Source: risks.jsonl:dynamic_include, risks.jsonl:sleep_in_loop
- The highest-complexity units must be decomposed into services with dedicated test coverage before cutover: inbox UI, webhook processEvent, appointments board and booking flow.
  Source: script:inbox/index.php@inbox/index.php:1, function:processEvent@includes/webhook_processor.php:8, script:appointments/index.php@appointments/index.php:1, function:handleBookingInbound@includes/booking_flow.php:660

Open questions

- Which PHP version does production run? (phpVersion is empty in the fact base.) — Version-specific semantics (type juggling, date handling, encryption functions) affect parity testing.
- No route usage telemetry was captured. Which routes are actually used in production? — Usage data is needed to prioritise or retire low-traffic pages such as help articles and diag/fix tools.

2. Business requirements

Scope covers the WA Blast Pro PHP application (MEDIUM size, 26,436 PHP LOC, plain PHP, PHP version not captured) across 12 modules: common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas and mobileapi. Requirements below are limited to facts present in the fact base.

Modules: each module maps to a Java package under `com.wablastpro.*`.

Roles and access: auth helpers (login, logout, role checks, CSRF, notifications, activity logging) are captured as units in `includes/auth.php`. The `users` table holds roles `admin`, `manager` and `viewer`; SaaS operators are held separately in `saas_admins` with roles `superadmin` and `support`.

Workflows: 66 captured journeys with 660 recorded fixtures are available to drive behavioural parity testing.

Business rules: the highest-complexity rule units include the WhatsApp booking flow (`bookingHandleTap`, `handleBookingInbound`), webhook event processing (`processEvent`), flow enrollment processing (`processDueFlowEnrollments`), plan limits (`checkPlanLimit`) and appointment reminders. These must be preserved in the target.

Validations: CSRF validation, GSTIN validation, DOB validation and setup-tool gating are captured.

Integrations: WhatsApp API client (`WhatsAppAPI`), Razorpay checkout and webhook (signature verification, subscriptions, orders), and the inbound WhatsApp webhook endpoints.

Scheduled jobs: nine cron scripts cover billing checks, flow processing, tier checks, appointment reminders, birthday wishes and campaign sending.

Redesign choices recorded in the plan: tenant bootstrap to Hibernate multitenancy, auto-reply processing to a queue, flow enrollments to delayed jobs, campaign sending to job queue or message queue, and inbox polling to websocket.

Risks to carry into requirements: 64 dynamic include sites, and sleep calls inside loops in campaign sending and auto-reply processing.

Requirements

- The system shall retain the 12 functional modules: common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas and mobileapi.
  Source: plan.modules:common, plan.modules:users, plan.modules:inbox, plan.modules:ai, plan.modules:leads, plan.modules:campaigns, plan.modules:flows, plan.modules:appointments, plan.modules:rewards, plan.modules:billing, plan.modules:saas, plan.modules:mobileapi
- The system shall authenticate users with login and logout and enforce role checks for protected functions.
  Source: function:login@includes/auth.php:40, function:logout@includes/auth.php:67, function:hasRole@includes/auth.php:34, function:requireRole@includes/auth.php:26
- The system shall store workspace users with roles admin, manager and viewer, and SaaS operators with roles superadmin and support.
  Source: table:users, table:saas_admins
- The system shall validate CSRF tokens on state-changing requests.
  Source: function:verifyCsrf@includes/auth.php:79, function:csrfToken@includes/auth.php:74, function:wablast_csrf_ok@config/env.php:140
- The system shall validate GSTIN values.
  Source: function:gstinValid@includes/gst.php:31
- The system shall validate date of birth values for birthday processing.
  Source: function:validateDob@includes/birthday.php:64
- The system shall enforce plan limits per tenant plan.

Source: function:checkPlanLimit@config/tenant.php:164, table:plans

- The system shall run the WhatsApp self-booking flow, including tap and inbound handling, to create appointments.
  Source: function:bookingHandleTap@includes/booking_flow.php:828, function:handleBookingInbound@includes/booking_flow.php:660
- The system shall process inbound WhatsApp webhook events and apply auto-replies.
  Source: function:processEvent@includes/webhook_processor.php:8, function:processAutoReply@includes/webhook_processor.php:361
- The system shall process due drip-flow enrollments according to step timing.
  Source: function:processDueFlowEnrollments@includes/flow_engine.php:6, table:flow_enrollments
- The system shall send appointment reminders on a schedule and record each send attempt.
  Source: function:sendAppointmentReminderForRow@includes/appointments.php:342, script:cron/send_appointment_reminders.php@cron/send_appointment_reminders.php:1
- The system shall send campaign messages through scheduled jobs and record per-recipient status.
  Source: script:cron/send_campaign.php@cron/send_campaign.php:1, script:cron/send_campaign_saas.php@cron/send_campaign_saas.php:1, table:campaigns, table:message_logs
- The system shall send birthday wishes on a schedule and record each send.
  Source: script:cron/send_birthday_wishes.php@cron/send_birthday_wishes.php:1, function:sendBirthdayWishForLead@includes/birthday.php:249, table:birthday_wish_sends
- The system shall run billing checks and accept Razorpay webhooks after verifying their signature.
  Source: script:cron/billing_check.php@cron/billing_check.php:1, script:billing/webhook.php@billing/webhook.php:1, function:razorpayVerifyWebhookSignature@includes/razorpay.php:79
- The system shall create Razorpay orders and subscriptions and record invoices.
  Source: function:razorpayCreateOrder@includes/razorpay.php:51, function:razorpayCreateSubscription@includes/razorpay.php:61, table:invoices
- The system shall issue invoices for billing events.
  Source: function:issueInvoice@includes/billing.php:75
- The system shall authenticate mobile API calls with bearer tokens, refresh sessions and support token revocation.
  Source: function:mobileRequireBearer@includes/mobile_api.php:83, table:mobile_device_sessions, table:mobile_access_revoke
- The system shall make mobile message sends idempotent using a stored idempotency key.
  Source: table:mobile_send_idempotency, script:api/mobile/v1/send.php@api/mobile/v1/send.php:1
- The system shall resolve and bootstrap the tenant for each request.
  Source: function:bootstrapTenant@config/tenant.php:288, function:detectTenant@config/tenant.php:48, table:tenants
- The system shall manage leads with opt-in and opt-out state and record opt events.
  Source: table:leads, table:opt_outs, table:opt_events
- The system shall manage appointments, services and staff time off.
  Source: table:appointments, table:services, table:staff_time_off
- The system shall manage reward programs, customer points balances and reward transactions.
  Source: table:reward_programs, table:customer_rewards, table:reward_transactions
- The system shall manage referral codes and referral payouts.

Source: table:referral_codes, table:referral_payouts

- The system shall manage drip flows, their steps and enrollments.
  Source: table:flows, table:flow_steps, table:flow_enrollments
- The system shall run AI chat sessions with a knowledge base and record handovers to humans.
  Source: table:ai_sessions, table:ai_handovers, table:knowledge_base
- The system shall provide a shared inbox with conversations, messages and replies.
  Source: table:conversations, table:chat_messages, table:chat_replies
- The system shall preserve the sleep-based throttling in campaign sending and auto-reply processing, reworked as queue rate limits in the target.
  Source: risks.jsonl:sleep_in_loop, cron/send_campaign.php:155, cron/send_campaign_saas.php:180, includes/webhook_processor.php:383

Open questions

- What is the PHP version in use? — phpVersion is blank in the fact base, which affects upgrade path and language-feature parity.
- Are the workflow fixtures and captured journeys mapped to specific business requirements? — Fixture counts are listed, but no requirement-to-journey traceability is captured.
- Which reports exist and what do they contain? — The fact base lists campaign reports in the module summary but no report units or outputs.
- Which role permissions apply to each route? — Routes are captured without an access-control mapping per route.
- What are the business rules encoded in the tier, referral and birthday reward logic? — Rewards rules are listed only by unit name and complexity, not by behaviour.
- Is the database logic list empty because stored procedures or triggers are absent, or because they were not captured? — databaseLogic is empty and openGaps is empty, so completeness is unclear.

3. Data model

The fact base lists 60+ MySQL tables across modules. Key groups:

- **Identity and access:** `users` (role enum admin/manager/viewer), `activity_log`, `notifications`, `mobile_device_sessions`, `mobile_access_revoke`, `mobile_send_idempotency`, `saas_admins`, `saas_settings`.
- **Tenancy and billing:** `tenants` (holds per-tenant `db_name` and encrypted `db_pass`), `plans`, `invoices`, `usage_logs`, `support_tickets`.
- **CRM and leads:** `leads`, `lead_groups`, `lead_group_members`, `lead_assignments`, `lead_tag_library`, `opt_outs`, `opt_out_keywords`, `opt_in_keywords`, `opt_in_links`, `opt_events`.
- **Messaging and inbox:** `conversations`, `chat_messages`, `chat_replies`, `inbox_messages`, `media_cache`, `webhook_logs`, `auto_replies`, `quick_replies`, `booking_sessions`.
- **Campaigns and templates:** `campaigns`, `templates`, `message_logs`.
- **Flows:** `flows`, `flow_steps`, `flow_enrollments`.
- **AI:** `ai_sessions`, `ai_conversations`, `ai_handovers`, `knowledge_base`.
- **Appointments:** `appointments`, `services`, `staff_time_off`, `appointment_reminder_sends`.

- **Rewards:** `reward_programs`, `reward_transactions`, `customer_rewards`, `referrals`, `referral_codes`, `referral_events`, `referral_payouts`, `birthday_wish_sends`.
- **Configuration:** `settings`, `saas_settings`.

Tenant isolation is handled at runtime by `bootstrapTenant` and `connectTenantDB` (config/tenant.php), which the plan redesigns to hibernate-multitenancy.

Note: `databaseLogic` in the fact base is empty, so table-level constraints, foreign keys and indexes are not captured.

Requirements

- The target data model must retain the core identity, tenant and audit tables (users, tenants, activity_log) with their existing columns.
  Source: table:users, table:tenants, table:activity_log
- Tenant database resolution currently in bootstrapTenant must be replaced by a hibernate-multitenancy approach as planned.
  Source: config/tenant.php:288, redesignChoices:function:bootstrapTenant
- Per-tenant database passwords stored in tenants.db_pass must remain encrypted at rest and be decrypted only by the tenant resolution layer.
  Source: table:tenants, function:encryptDbPass@config/tenant.php:256, function:decryptDbPass@config/tenant.php:263
- Mobile API session, token revocation and idempotency tables must be preserved to keep refresh-token and idempotent send behaviour.
  Source: table:mobile_device_sessions, table:mobile_access_revoke, table:mobile_send_idempotency
- Billing data (plans, invoices, usage) must be migrated with amount, currency and period fields intact.
  Source: table:invoices, table:plans, table:usage_logs
- Campaign, template and delivery-log tables must be migrated so existing campaign reports remain reproducible.
  Source: table:campaigns, table:message_logs, table:templates
- Flow definitions, steps and enrollment state must be migrated so in-progress drip enrollments can resume after cutover.
  Source: table:flow_enrollments, table:flow_steps, table:flows
- Appointment, service and booking-session tables must be migrated with soft-delete (deleted_at) semantics preserved.
  Source: table:appointments, table:services, table:booking_sessions
- Lead and opt-in/opt-out tables must be migrated, keeping opt-out history for compliance.
  Source: table:leads, table:opt_outs, table:opt_events
- Webhook payload log and media cache tables must be retained to support replay and media proxy behaviour.
  Source: table:webhook_logs, table:media_cache

Open questions

- Are foreign keys and indexes defined in the schema, and are they enforced? — The fact base lists columns only; `databaseLogic` is empty, so referential integrity is unknown.

- Which tables are shared across tenants versus created per tenant database? — Tenancy resolution uses per-tenant db_name, but the fact base does not map each table to master or tenant DB.
- Is `conversations`/`chat_messages` duplicated by `inbox_messages`, and which is authoritative? — Overlapping inbox-related tables appear in the fact base with no ownership stated.
- How are `db_pass` values encrypted and where is the key held? — `tenants.db_pass` is stored, and encryptDbPass/decryptDbPass exist, but key management is not documented.
- Is `auto_replies.regex` and similar free-text field validated before use? — The fact base shows a regex trigger type but no validation rule.

4. Technical assessment

- Codebase: WA Blast Pro, plain PHP (no framework), 26,436 PHP lines, sizeBand MEDIUM.
- PHP version is not recorded in the fact base; treat the source runtime version as unknown until confirmed.
- Scope spans 12 modules (common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas, mobileapi) mapped to Java packages com.wablastpro.*.
- Routing is file-based: roughly 100+ direct PHP script routes (for example /inbox/index.php, /billing/webhook.php, /api/mobile/v1/send.php), plus a v2 mobile entry point at /api/mobile/v2/index.php.
- Data layer: about 60 MySQL tables, including multi-tenant concepts (tenants, usage_logs, invoices, saas_settings) and mobile auth tables (mobile_device_sessions, mobile_access_revoke, mobile_send_idempotency).
- Multi-tenancy is per-tenant database: tenants carry db_name and db_pass, and bootstrapTenant is redesigned to hibernate-multitenancy.
- Highest complexity hotspots: `WhatsAppAPI` class (complexity 189), processEvent (complexity 134), inbox/index.php (complexity 192, 1028 LOC), appointments/index.php (complexity 120), inbox/logs.php (complexity 113), handleBookingInbound (complexity 81).
- Risk register: 64 dynamic_include findings (include with computed path) and 3 sleep_in_loop findings (cron/send_campaign.php:155, cron/send_campaign_saas.php:180, includes/webhook_processor.php:383). Dynamic includes are a Java-migration blocker and must be replaced with explicit class wiring.
- Scheduled work: 10 cron scripts, with planned redesigns to delayed-jobs, job-queue, message-queue, jobrunr, batch and event-driven patterns.
- Security-relevant surfaces: setup/diagnostic tools (diag.php, fix.php, scripts/create_saas_admin.php) and deploy/caddy-ask.php must be restricted or removed in the target.
- Approved plan waves (corrected fact base): Wave 1 — 145 units (4713 LOC, 48 routes); Wave 2 — 98 units (7870 LOC, 35 routes); Wave 3 — 93 units (7134 LOC, 26 routes); Wave 4 — 146 units (5974 LOC, 25 routes); Wave 5 — 76 units (3758 LOC, 5 routes); Wave 6 — 35 units (1366 LOC). Total 593 units across six waves. Sequencing is defined by the approved plan; confirm effort and order with the tech lead.

- Runtime: date.timezone is set to Asia/Kolkata in code and must be preserved.
- Captured fixtures: 660 recorded fixtures across 66 journeys, usable as regression baseline.

Requirements

- The target Java 21 system must cover all 12 identified modules mapped to com.wablastpro packages.
  Source: plan.modules:common, plan.modules:users, plan.modules:inbox, plan.modules:ai, plan.modules:leads, plan.modules:campaigns, plan.modules:flows, plan.modules:appointments, plan.modules:rewards, plan.modules:billing, plan.modules:saas, plan.modules:mobileapi
- Computed-path includes (64 occurrences) must be replaced with explicit, statically resolvable Java wiring.
  Source: risks.jsonl:dynamic_include, api/mobile/v1/auth/login.php:2, cron/billing_check.php:12
- Blocking sleep and usleep calls inside loops must be replaced with non-blocking scheduling or rate-limited queue consumers.
  Source: cron/send_campaign.php:155, cron/send_campaign_saas.php:180, includes/webhook_processor.php:383, risks.jsonl:sleep_in_loop
- The WhatsApp client, webhook event processing and shared inbox are the highest complexity components and require dedicated refactoring and test coverage before cutover.
  Source: class:WhatsAppAPI@config/whatsapp.php:5, function:processEvent@includes/webhook_processor.php:8, script:inbox/index.php@inbox/index.php:1
- Tenant resolution must be redesigned to hibernate-multitenancy, replacing per-request DB connection bootstrapping.
  Source: function:bootstrapTenant@config/tenant.php:288, plan.redesignChoices:function:bootstrapTenant
- Scheduled cron jobs for campaigns, reminders, flows and billing must be migrated to the planned queue, delayed-job or jobrunr mechanisms.
  Source: plan.redesignChoices:script:cron/send_campaign.php, plan.redesignChoices:script:cron/send_campaign_saas.php, plan.redesignChoices:script:cron/send_appointment_reminders.php, plan.redesignChoices:script:cron/process_flows_saas.php
- The application timezone Asia/Kolkata must be preserved in the Java runtime configuration.
  Source: runtime.json:date.timezone, script:cron/run_tier_checks.php@cron/run_tier_checks.php:1
- The 660 recorded fixtures across 66 journeys must be used as the parity regression baseline for the Java implementation.
  Source: capture:bb058f1f-c343-445f-aa22-a1e9257fb203, capture.journey:1
- Setup and diagnostic tools exposed as routes must be restricted or excluded from the target deployment.
  Source: route:ANY /diag.php@diag.php:1, route:ANY /fix.php@fix.php:1, script:scripts/create_saas_admin.php

Open questions

- What is the source PHP version and which extensions are required? — phpVersion is empty in the fact base, which blocks build and behaviour parity planning.
- How will the 64 dynamic includes be resolved and validated? — Computed include paths are a major risk and their target class mapping is not in the fact base.

- Is the per-tenant database model retained in the Java target? — The redesign choice for bootstrapTenant is hibernate-multitenancy, but schema isolation and migration of tenant DBs are not specified.
- Are the cron-to-queue redesigns approved, and what are the delivery guarantees required? — Campaign and reminder sending must preserve idempotency and rate limiting (usleep loops) in the new design.
- Should the setup and diagnostic tools (diag.php, fix.php) be removed or retained behind controls? — These tools are exposed routes and represent a security exposure if carried over.

5. Target architecture

Shape: a Java 21 modular monolith, with one package per planned module (`com.wablastpro.<module>`).

Multi-tenancy

- **Tenant resolution:** `bootstrapTenant` maps to Hibernate multi-tenancy (database-per-tenant). Tenants are resolved by:
  – slug
  – custom domain
  – WhatsApp phone number
  – tenant id
- **Database layout:** a master DB holds tenants, plans, invoices, usage and SaaS admins. Tenant DB credentials are encrypted at rest.
- **Usage controls:** plan limits and usage metering (`checkPlanLimit`, `incrementUsage`) become a shared billing-policy service.

Async & scheduling (from plan redesign choices)

Legacy	Target
processAutoReply	queue
processDueFlowEnrollments	delayed-jobs
cron/process_flows_saas.php	JobRunr
cron/send_campaign.php	job-queue
cron/send_campaign_saas.php	message-queue
cron/send_appointment_reminders.php	delayed-jobs
cron/send_birthday_wishes.php	batch
cron/billing_check.php	event-driven
cron/run_tier_checks.php	event-driven
inbox/poll.php	WebSocket

Security

- **Web sessions:**
  – secure session cookies
  – CSRF tokens
  – login throttling
  – workspace roles `admin`, `manager` and `viewer`
- **Operator console:** separate `superadmin` and `support` roles, plus an IP allow-list.
- **Mobile API:**
  – JWT bearer tokens
  – hashed refresh tokens per device
  – access-token revocation by `jti`
  – idempotency keys for send endpoints
- **Path protection:** `.htaccess` blocks direct access to `config/`, `cron/`, `includes/` and `database/`. Equivalent protection must exist in Java (non-routable internals).

Cross-cutting

- Default timezone **Asia/Kolkata**.
- Environment-driven config (`envv`, `env_bool`, `app_env`, `is_production`).
- Setup tools are gated by `allow_setup_tools`.
- GST invoicing with GSTIN validation.

Requirements

- Database-per-tenant isolation must be implemented with Hibernate multi-tenancy. A master datasource must hold tenant metadata.
  Source: plan.redesignChoices:function:bootstrapTenant, function:connectTenantDB@config/tenant.php:106, function:getMasterDB@config/tenant.php:32, table:tenants
- Tenant resolution must support lookup by slug, custom domain, WhatsApp phone number and tenant id. WhatsApp phone lookup is needed for webhook routing.
  Source: function:detectTenant@config/tenant.php:48, function:getTenantBySlug@config/tenant.php:68, function:getTenantByCustomDomain@config/tenant.php:91, function:getTenantByWAPhone@config/tenant.php:98, function:getTenantById@config/tenant.php:77
- Tenant DB credentials must remain encrypted at rest. Decryption must stay compatible with existing stored values during cutover.
  Source: function:encryptDbPass@config/tenant.php:256, function:decryptDbPass@config/tenant.php:263
- Plan-limit enforcement and monthly usage metering must be centralised in a billing-policy service. Metered values are campaigns_sent, ai_replies, contacts_added and messages_sent.
  Source: function:checkPlanLimit@config/tenant.php:164, function:incrementUsage@config/tenant.php:214, function:getUsage@config/tenant.php:228, table:usage_logs, table:plans
- Auto-reply processing must move to a queue consumer. This removes the blocking sleep() from the webhook path.
  Source: plan.redesignChoices:function:processAutoReply, includes/webhook_processor.php:383
- Campaign sending must run on a job/message queue with explicit rate limiting. This replaces usleep loops.

Source: plan.redesignChoices:script:cron/send_campaign.php, plan.redesignChoices:script:cron/send_campaign_saas.php, cron/send_campaign.php:155, cron/send_campaign_saas.php:180

- Flow enrollments must be executed as delayed jobs on JobRunr, scheduled from flow_enrollments.next_send_at.
  Source: plan.redesignChoices:function:processDueFlowEnrollments, plan.redesignChoices:script:cron/process_flows_saas.php, table:flow_enrollments
- Scheduled work must be redesigned as follows: appointment reminders as delayed jobs, birthday wishes as batch, and billing checks and tier checks as event-driven handlers.
  Source: plan.redesignChoices:script:cron/send_appointment_reminders.php, plan.redesignChoices:script:cron/send_birthday_wishes.php, plan.redesignChoices:script:cron/billing_check.php, plan.redesignChoices:script:cron/run_tier_checks.php
- Inbox polling must be replaced by WebSocket push for new messages and unread counts.
  Source: plan.redesignChoices:script:inbox/poll.php, route:ANY /inbox/poll.php@inbox/poll.php:1
- The mobile API must keep all of the following: JWT bearer auth, hashed per-device refresh tokens, jti-based access revocation and idempotency-key handling for sends.
  Source: function:mobileRequireBearer@includes/mobile_api.php:83, table:mobile_device_sessions, table:mobile_access_revoke, table:mobile_send_idempotency
- Web security must preserve CSRF protection, secure session configuration and login-attempt throttling.
  Source: function:verifyCsrf@includes/auth.php:79, function:wablast_csrf_ok@config/env.php:140, function:startSecureSession@config/env.php:98, function:wablast_login_blocked@config/env.php:145, function:wablast_login_fail@config/env.php:159
- Role-based access must enforce workspace roles (admin, manager, viewer). It must also enforce operator roles (superadmin, support), with an IP allow-list for the operator console.
  Source: table:users, function:requireRole@includes/auth.php:26, table:saas_admins, function:saasAdminIpMatches@includes/saas_admin.php:7
- Internal code must not be reachable over HTTP. This matches the .htaccess forbidden rules for config/, cron/, includes/ and database/.
  Source: urls.json
- The application default timezone must be Asia/Kolkata, so that date and scheduling behaviour matches the legacy system.
  Source: runtime.json
- Setup and diagnostic capabilities must be disabled by default. They may only be enabled via explicit configuration, and never in production unless explicitly allowed.
  Source: function:allow_setup_tools@config/env.php:64, function:deny_unless_setup_tools@config/env.php:122, function:is_production@config/env.php:53
- GST invoice issuance and GSTIN validation must be ported with identical numbering and validation rules.
  Source: function:gstinValid@includes/gst.php:31, function:issueInvoice@includes/billing.php:75, table:invoices

Open questions

- Which Java web framework and which message broker or job runtime should be standardised? The plan mixes queue, job-queue, message-queue, delayed-jobs, batch and JobRunr. — A single async backbone reduces operational surface. The fact base names patterns but not a concrete broker.

- Should the single-tenant cron variants be retired in favour of the SaaS variants? These are send_campaign.php vs send_campaign_saas.php and process_flows.php vs process_flows_saas.php. — Duplicate execution paths double migration and test effort.
- Must mobile API v1 and v2 both be preserved, or can v1 be deprecated? — Both are routed today, and the answer drives API compatibility scope for the mobile app.
- Should diag.php, fix.php and deploy/caddy-ask.php be migrated, replaced by actuator/ops tooling, or dropped? — Exposed diagnostic and repair endpoints are a security risk. Caddy on-demand TLS 'ask' behaviour must be kept if custom domains rely on it.
- Which database engine and version is in use? Also, are inbox_messages and chat_messages both authoritative? — The schema (enum, tinyint, json) implies MySQL/MariaDB but this is not stated. The overlapping message tables affect data-model consolidation.
- Which LLM provider and model does AIEngine call, and what are the outbound rate and throughput constraints for WhatsApp campaign sends? — Provider choice and rate limits are needed to design the AI client and to replace the usleep-based throttling in the campaign senders.

6. Migration plan

- Approved plan waves (corrected fact base): Wave 1 — 145 units (4713 LOC, 48 routes); Wave 2 — 98 units (7870 LOC, 35 routes); Wave 3 — 93 units (7134 LOC, 26 routes); Wave 4 — 146 units (5974 LOC, 25 routes); Wave 5 — 76 units (3758 LOC, 5 routes); Wave 6 — 35 units (1366 LOC). Total 593 units across six waves.
  Scope: WA Blast Pro, MEDIUM size band, 26436 PHP lines, plain PHP framework, 12 modules (common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas, mobileapi). The plan wave allocation is empty, so sequencing below is a draft that needs confirmation.

Approach

- Map each module to a Java package (com.wablastpro.<module>) and migrate module by module, starting with common since it carries shared infrastructure.
- Replace the per-request tenant bootstrap with a multitenancy approach (hibernate-multitenancy).
- Move asynchronous and scheduled work to queues and job runners instead of PHP cron scripts and inline loops.
- Resolve dynamic includes and in-loop sleeps before porting the affected units.

Redesign decisions already recorded

- bootstrapTenant: hibernate-multitenancy.
- processAutoReply: queue.
- processDueFlowEnrollments: delayed-jobs.
- send_campaign.php: job-queue; send_campaign_saas.php: message-queue.
- send_appointment_reminders.php: delayed-jobs; send_birthday_wishes.php: batch.
- run_tier_checks.php and billing_check.php: event-driven; process_flows_saas.php: jobrunr.

- inbox/poll.php: websocket.

Risk hotspots to plan for

- Dynamic include risk count is 64, concentrated in api/mobile, cron, includes and rewards; each needs a static replacement or explicit mapping.
- Three sleep-in-loop occurrences (cron/send_campaign.php, cron/send_campaign_saas.php, includes/webhook_processor.php) must be replaced by scheduled or queued delays.
- Highest complexity units: `WhatsAppAPI` (189), inbox/index.php (192, 1028 LOC), processEvent (134), handleBookingInbound (81), `AIEngine` (90). These should be migrated later with dedicated test coverage.

Proposed sequencing (draft)

1. common (config, tenant, session, SQL safety helpers).
2. users and saas (auth and tenant provisioning).
3. billing and rewards.
4. leads, campaigns and flows (queue and job redesign).
5. inbox, ai and appointments (webhook-heavy paths).
6. mobileapi (JWT and idempotent send endpoints).

Validation relies on the 660 recorded fixtures and the 60+ capture journeys, which should be used as regression baselines.

Requirements

- Migrate the 12 modules as separate Java packages under com.wablastpro, with common migrated first.
  Source: plan.modules:common, plan.modules:users, plan.modules:inbox, plan.modules:ai, plan.modules:leads, plan.modules:campaigns, plan.modules:flows, plan.modules:appointments, plan.modules:rewards, plan.modules:billing, plan.modules:saas, plan.modules:mobileapi
- Replace the tenant bootstrap with hibernate-multitenancy.
  Source: plan.redesignChoices:function:bootstrapTenant
- Move auto-reply processing, flow enrollment processing and campaign sending from inline or cron execution to queues or delayed jobs.
  Source: plan.redesignChoices:function:processAutoReply, plan.redesignChoices:function:processDueFlowEnrollments, plan.redesignChoices:script:cron/send_campaign.php, plan.redesignChoices:script:cron/send_campaign_saas.php
- Replace the scheduled cron scripts with delayed jobs, batch, event-driven or jobrunr processing as recorded in the redesign choices.
  Source: plan.redesignChoices:script:cron/send_appointment_reminders.php, plan.redesignChoices:script:cron/send_birthday_wishes.php, plan.redesignChoices:script:cron/run_tier_checks.php, plan.redesignChoices:script:cron/billing_check.php, plan.redesignChoices:script:cron/process_flows_saas.php
- Replace the inbox polling endpoint with a websocket-based design.
  Source: plan.redesignChoices:script:inbox/poll.php
- Resolve the 64 dynamic include occurrences into explicit class or module references before porting the affected units.

Source: risks.jsonl:dynamic_include

- Replace the three sleep or usleep calls inside loops with scheduled or queued delays.
  Source: risks.jsonl:sleep_in_loop, cron/send_campaign.php:155, cron/send_campaign_saas.php:180, includes/webhook_processor.php:383
- Schedule the highest-complexity units (WhatsAppAPI, inbox index, processEvent, AIEngine) after their dependencies and with dedicated regression coverage.
  Source: class:WhatsAppAPI@config/whatsapp.php:5, script:inbox/index.php@inbox/index.php:1, function:processEvent@includes/webhook_processor.php:8, class:AIEngine@config/ai.php:9
- Use the 660 recorded capture fixtures and the journey set as the regression baseline for each migrated wave.
  Source: capture:bb058f1f-c343-445f-aa22-a1e9257fb203, capture.journey:1

Open questions

- Which PHP version is the source runtime, given the phpVersion field is empty? — Language-level migration gaps and library mappings depend on the source version.
- Is the mobile API v1 kept running in parallel with v2 during cutover, and for how long? — Mobile clients depend on JWT and refresh-token behaviour that must stay stable.
- What is the target deployment for the queue, job runner and websocket components? — Redesign choices name these components but the infrastructure target is not stated.
- Are the setup and diagnostic tools (diag.php, fix.php, scripts) to be migrated or retired? — They contain dynamic includes and are reachable routes that affect scope and security.

7. Risk register

Summary of risks recorded in the fact base for WA Blast Pro (MEDIUM, 26436 PHP LOC, PLAIN_PHP, PHP version not captured).

- **Dynamic include (64 occurrences):** includes with computed paths across mobile API bootstrap, cron jobs, appointments, birthday, help rendering, rewards and scripts. Each needs path-allowlisting review before porting, since Java has no equivalent runtime include.
- **Sleep inside loops (3 occurrences):** `usleep()` in campaign senders (`cron/send_campaign.php:155`, `cron/send_campaign_saas.php:180`) and `sleep()` in `processAutoReply` (`includes/webhook_processor.php:383`). Blocking loops conflict with the planned queue-based redesign.
- **Unresolved PHP version:** `phpVersion` is empty, so language-level compatibility risk is not yet quantified.
- **Dev/setup tooling exposed by routes:** `diag.php` and `fix.php` are routable and contain high complexity (44 and 19); these must be removed or gated in production.
- **Highly complex units:** `processEvent` (complexity 134), `WhatsAppAPI` (189), `inbox/index.php` (192, 1028 LOC) and `handleBookingInbound` (81) concentrate migration risk.
- **Open gaps:** `openGaps` and `databaseLogic` are empty in the fact base, so completeness of the risk picture is unverified.

Requirements

- Dynamic include paths must be reviewed and replaced with explicit Java class wiring during migration.
  Source: risks.jsonl:dynamic_include, api/mobile/v1/auth/login.php:2
- Blocking usleep loops in campaign senders must be replaced by queue-based sending in the target design.
  Source: risks.jsonl:sleep_in_loop, cron/send_campaign.php:155, cron/send_campaign_saas.php:180
- The blocking sleep inside processAutoReply must be removed and the auto-reply processing moved to an asynchronous queue.
  Source: includes/webhook_processor.php:383
- Diagnostic and fix scripts exposed as routes must be removed or restricted to non-production environments before go-live.
  Source: script:diag.php, script:fix.php, route:ANY /diag.php, route:ANY /fix.php
- The WhatsApp API client and webhook event processor are high-complexity units requiring focused test coverage before migration.
  Source: class:WhatsAppAPI@config/whatsapp.php:5, function:processEvent@includes/webhook_processor.php:8
- The source PHP version must be confirmed before the Java 21 migration risk can be finalised.
  Source: phpVersion

Open questions

- What is the PHP runtime version in use? — phpVersion is empty in the fact base and blocks compatibility assessment.
- Are diag.php and fix.php reachable in production? — Both are routed and carry setup/diagnostic logic; exposure is not confirmed.
- Which of the 64 dynamic includes receive user-controlled input? — Computed include paths can be a code-execution risk; the fact base lists locations but not inputs.

8. Test and parity strategy

- Capture route usage (fixture routeKeys): 164 distinct routes across 660 fixtures (for example login.php appears 130 times). Access-log intake (P04b) was not imported, so production hit counts may differ from the capture.
- The capture holds 660 recorded fixtures, which can serve as the PHP-to-Java parity baseline.
- The largest journey (journey 1) has 289 fixtures; other journeys have 2 to 13 each.
- Parity gating should follow the approved plan waves (593 units across six waves — see migration plan). Confirm which recorded journeys attach to each wave before run.
- Scheduled jobs, the timezone setting, and the redesigned async components need targeted parity checks.
- Dynamic includes and sleep-in-loop patterns are recorded risks that parity tests should exercise.

Requirements

- Use the 660 recorded capture fixtures as the baseline for PHP-to-Java parity testing.
  Source: capture:bb058f1f-c343-445f-aa22-a1e9257fb203
- Prioritise parity tests for journey 1, which has 289 fixtures and is the largest recorded journey.
  Source: capture.journey:1
- Define parity exit criteria per migration wave before cutover, since current waves contain no units.
  Source: plan.waves[0], plan.waves[1], plan.waves[2], plan.waves[3], plan.waves[4], plan.waves[5]
- Verify campaign sending parity, including the move from cron to a job queue, with tests covering both send_campaign and send_campaign_saas.
  Source: script:cron/send_campaign.php@cron/send_campaign.php:1, script:cron/send_campaign_saas.php@cron/send_campaign_saas.php:1, plan.redesignChoices:script:cron/send_campaign.php
- Include timing-sensitive tests for the usleep loops in campaign sending, since these are flagged as sleep-in-loop risks.
  Source: risks.jsonl:sleep_in_loop, cron/send_campaign.php:155, cron/send_campaign_saas.php:180
- Test auto-reply timing parity around the sleep call in processAutoReply, given its planned change to queue-based processing.
  Source: includes/webhook_processor.php:383, plan.redesignChoices:function:processAutoReply
- Add tests for each computed-path include in cron and reminder scripts, since 64 dynamic includes are recorded.
  Source: risks.jsonl:dynamic_include, cron/send_appointment_reminders.php:17, cron/send_birthday_wishes.php:15
- Verify that scheduled reminders run in the Asia/Kolkata timezone after moving to delayed jobs.
  Source: runtime.json, plan.redesignChoices:script:cron/send_appointment_reminders.php

Open questions

- Does the capture's 164 distinct routes (660 fixtures) cover the production-critical paths, or should access-log intake be imported to refine priorities? — Fixture routeKeys now populate routeUsage; production access-log intake was not imported, so low-traffic or uncaptured routes may still need stakeholder confirmation.
- What is the PHP version to target for parity runs? — phpVersion is blank in the fact base.
- Does an existing automated test suite exist beyond the captured fixtures? — The fact base records fixtures and risks but no test suite or coverage data.
- Are there any open gaps that block parity sign-off? — openGaps is empty; it is unclear whether this means no gaps or an incomplete analysis.

9. Database plan

Database migration plan for WA Blast Pro (PHP to Java 21). The fact base lists 50+ tables across the tenant schema, including core tables such as leads, conversations, chat_messages, campaigns, message_logs, flows, flow_enrollments, appointments, booking_sessions, customer_rewards, invoices, tenants, users and webhook_logs. Table column detail is captured per table. Multi-tenancy is per-tenant database (tenants.db_name, db_pass) and the plan redesigns bootstrapTenant to hibernate-multitenancy. No database logic units were

captured (databaseLogic is empty), so stored procedures, triggers and query-level logic are unverified. Recommended approach: map each captured table to a JPA entity with explicit enum mappings, keep the existing column names for the cutover, and verify tenant-DB routing before any data move.

Requirements

- Tenant database routing must be redesigned for Java using per-tenant connection metadata from the tenants table (db_name, db_pass) before data migration.
  Source: table:tenants, plan.redesignChoices:function:bootstrapTenant
- Core CRM and messaging tables (leads, message_logs, chat_messages) must be migrated with their enum status and direction domains preserved.
  Source: table:leads, table:message_logs, table:chat_messages
- Flow automation tables (flows, flow_steps, flow_enrollments) must be migrated together to keep enrollment state and step ordering consistent.
  Source: table:flow_enrollments, table:flows, table:flow_steps
- Billing tables (invoices, tenants) hold amounts in integer and Razorpay identifiers and must be migrated with exact value and identifier fidelity.
  Source: table:invoices, table:tenants
- Mobile API tables for refresh sessions, idempotency keys and revoked JWT ids must be migrated or reset deliberately, since they affect token validity and send deduplication.
  Source: table:mobile_device_sessions, table:mobile_send_idempotency, table:mobile_access_revoke
- Appointment self-booking state (booking_sessions) and reminder send history must be migrated so in-flight bookings and reminder dedupe are preserved.
  Source: table:booking_sessions, table:appointments, table:appointment_reminder_sends
- The database logic inventory is empty in the fact base, so the plan must include a separate verification step for any query-level or stored logic before cutover.
  Source: databaseLogic:empty, runtime.json

Open questions

- Are there stored procedures, triggers or views in the tenant or master schemas? — databaseLogic is empty in the fact base, so non-table database behaviour cannot be scoped.
- What is the MySQL/MariaDB version and character set of the existing tenant databases? — Required for correct JSON, decimal and utf8mb4 mapping; not captured in the fact base (phpVersion is also empty).
- Is a schema migration tool (Flyway/Liquibase) already in use, or are schema changes applied by PHP scripts such as scripts/apply_mobile_api_patch.php? — Determines how the baseline schema is captured and versioned for the Java target.
- Which tables hold personal data (leads phone/email, users email, tenants owner_email) and what retention applies? — Needed to plan encryption, masking and deletion handling during migration.

10. Open questions

- The PHP runtime version is not captured (`phpVersion` is empty), so the baseline for the Java 21 migration is unknown.

- Approved plan waves (corrected fact base): Wave 1 — 145 units (4713 LOC, 48 routes); Wave 2 — 98 units (7870 LOC, 35 routes); Wave 3 — 93 units (7134 LOC, 26 routes); Wave 4 — 146 units (5974 LOC, 25 routes); Wave 5 — 76 units (3758 LOC, 5 routes); Wave 6 — 35 units (1366 LOC). Total 593 units across six waves.
- Database logic, intake jobs and encoded files are empty in the fact base (may mean none exist or intake was incomplete). Capture route usage (fixture routeKeys): 164 distinct routes across 660 fixtures (for example login.php appears 130 times). Access-log intake (P04b) was not imported, so production hit counts may differ from the capture.
- There are 64 dynamic include risks and 3 sleep-in-loop risks. Their runtime behaviour and target handling are not yet confirmed.
- Several setup and diagnostic endpoints (diag.php, fix.php, scripts/apply_mobile_api_patch.php, scripts/create_saas_admin.php) are present. Their intended production exposure is not stated.
- Role and permission model details beyond the auth helper functions (admin, manager, viewer and SaaS superadmin/support) are not documented in the fact base.
- Redesign choices are proposed for several cron jobs and functions, but acceptance criteria and owners are not defined.
- Several high-complexity units (`WhatsAppAPI`, processEvent, handleBookingInbound, inbox/index.php) lack captured behavioural expectations beyond the 660 recorded fixtures.

Open questions

- What PHP version is the source system running? — phpVersion is empty in the fact base, which blocks migration baseline and compatibility analysis.
- Do database logic, intake jobs and encoded files truly contain no items? — These collections are empty in the fact base and may reflect a capture/intake gap. Route usage is no longer empty — it is derived from capture fixture routeKeys (164 distinct routes).
- How will the 64 dynamic includes be resolved in Java? — Computed include paths cannot be translated directly and affect routing and module boundaries.
- Should diag.php, fix.php and the patch/admin scripts be retained, removed or restricted in production? — These setup and diagnostic tools expose operational risk and need an explicit disposition.
- How should the 3 usleep/sleep calls inside loops be replaced? — Blocking sleeps in campaign sending and auto-reply paths affect throughput and the planned queue redesign.
- What are the acceptance criteria for the proposed redesigns (queue, delayed-jobs, jobrunr, websocket)? — Redesign choices are listed without success measures, so parity cannot be verified.
- What is the full role and permission matrix for tenant users and SaaS operators? — Only auth helper functions and role-related routes are captured, not the rules they enforce.
- Who owns validation of the 660 recorded fixtures against the Java target? — Fixtures are the main behavioural baseline for high-complexity units such as WhatsAppAPI and processEvent.

11. Stakeholder interview guide

Use this guide with business owners, operators and engineering leads to confirm the facts captured for the MEDIUM-sized plain PHP codebase (26436 PHP lines) before the migration plan is finalised.

1. Scope and modules

- Confirm the 12 modules (common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas, mobileapi) and which are in scope for the first migration wave.
- Ask which modules are actively used by tenants and which are legacy or dormant.

2. Roles and access

- Confirm the workspace roles (admin, manager, viewer) and the SaaS operator roles (superadmin, support).
- Ask how permissions are expected to differ in the Java target, and whether the mobile API must honour the same role model.

3. Business workflows

- Walk through the highest-volume journeys (campaign send, shared inbox reply, appointment self-booking, flow enrollment, Razorpay checkout).
- Ask which workflows must keep identical behaviour and which can be simplified.

4. Business rules and validations

- Review plan limits, trial expiry, GST/GSTIN validation, date-of-birth validation, CSRF and login lockout rules.
- Ask for the authoritative source of truth for each rule where code and documentation may differ.

5. Integrations

- Confirm the WhatsApp Cloud API webhook and send paths, Razorpay checkout and webhook signature handling, and the AI engine provider.
- Ask about credentials ownership and rotation, since secrets were redacted in the fact base.

6. Scheduled jobs and async work

- Confirm cron schedules for campaign sending, flow processing, appointment reminders, birthday wishes, billing checks and tier checks.
- Ask about acceptable latency and failure handling for each job, and the proposed redesign targets.

7. Data and multi-tenancy

- Confirm the tenant isolation model (per-tenant database versus shared master) and the retention rules for leads, trash and activity logs.
- Ask about data migration tolerance and cutover windows.

8. Operations and risk

- Review the dynamic include and sleep-in-loop risks, and the diagnostic and fix scripts exposed on the web root.
- Ask who owns the setup tools and whether they must be removed or protected in production.

9. Non-functional requirements

- Confirm expected concurrency, webhook throughput, timezone handling (Asia/Kolkata), and availability targets.

10. Acceptance

- Agree the sign-off criteria for each wave and who validates captured journeys against the Java implementation.

Requirements

- Interview sessions must confirm scope and migration priority for all 12 modules.
  Source: plan.modules:common, plan.modules:users, plan.modules:inbox, plan.modules:ai, plan.modules:leads, plan.modules:campaigns, plan.modules:flows, plan.modules:appointments, plan.modules:rewards, plan.modules:billing, plan.modules:saas, plan.modules:mobileapi
- Confirm role-based access rules as implemented by requireRole and hasRole before designing Java authorization.
  Source: function:requireRole@includes/auth.php:26, function:hasRole@includes/auth.php:34
- Validate plan limit and trial expiry business rules with stakeholders.
  Source: function:checkPlanLimit@config/tenant.php:164, function:isTrialExpired@config/tenant.php:249
- Validate date-of-birth and GSTIN validation rules with stakeholders.
  Source: function:validateDob@includes/birthday.php:64, function:gstinValid@includes/gst.php:31
- Confirm scheduled job cadence, ownership and failure handling for each cron script.
  Source: script:cron/send_campaign.php@cron/send_campaign.php:1, script:cron/send_appointment_reminders.php@cron/send_appointment_reminders.php:1, script:cron/process_flows_saas.php@cron/process_flows_saas.php:1, script:cron/send_birthday_wishes.php@cron/send_birthday_wishes.php:1, script:cron/billing_check.php@cron/billing_check.php:1
- Confirm Razorpay and WhatsApp integration ownership, credentials handling and webhook signature verification requirements.
  Source: script:config/razorpay.php@config/razorpay.php:1, function:razorpayVerifyWebhookSignature@includes/razorpay.php:79, script:config/whatsapp.php@config/whatsapp.php:1
- Review the dynamic include and sleep-in-loop risk inventory with engineering to agree remediation priorities.
  Source: risks.jsonl:dynamic_include, risks.jsonl:sleep_in_loop

- Confirm the timezone setting (Asia/Kolkata) is required in all migrated scheduled and reporting logic.
  Source: runtime.json

Open questions

- Which PHP version is currently running in production? — The fact base leaves phpVersion empty, which affects the Java migration tooling and compatibility checks.
- Which tenant isolation model is in production: per-tenant databases or a shared schema? — Tenant bootstrap and DB connection logic will be redesigned differently for each model.
- Are the diagnostic and fix scripts (diag.php, fix.php) still reachable in production, and should they be retired? — These scripts carry high-complexity logic and dynamic includes and expose operational risk.

12. Open questions register

Question	Why it matters	Who can answer	Blocks which wave
Which PHP version does production run? (phpVersion is empty in the fact base.)	Version-specific semantics (type juggling, date handling, encryption functions) affect parity testing.	Platform / ops	Wave 1
No route usage telemetry was captured. Which routes are actually used in production?	Usage data is needed to prioritise or retire low-traffic pages such as help articles and diag/fix tools.	Sponsor + tech lead	TBD
What is the PHP version in use?	phpVersion is blank in the fact base, which affects upgrade path and language-feature parity.	Platform / ops	Wave 1
Are the workflow fixtures and captured journeys mapped to specific business requirements?	Fixture counts are listed, but no requirement-to-journey traceability is captured.	Sponsor + tech lead	TBD
Which reports exist and what do they contain?	The fact base lists campaign reports in the module summary but no report units or outputs.	Sponsor + tech lead	TBD
Which role permissions apply to each route?	Routes are captured without an access-control mapping per route.	Product owner	TBD
What are the business rules encoded in the tier, referral and birthday reward logic?	Rewards rules are listed only by unit name and complexity, not by behaviour.	Sponsor + tech lead	TBD
Is the database logic list empty because stored procedures or triggers are absent, or because they were not captured?	databaseLogic is empty and openGaps is empty, so completeness is unclear.	Sponsor + tech lead	TBD

Question	Why it matters	Who can answer	Blocks which wave
Are foreign keys and indexes defined in the schema, and are they enforced?	The fact base lists columns only; `databaseLogic` is empty, so referential integrity is unknown.	DBA / backend lead	TBD
Which tables are shared across tenants versus created per tenant database?	Tenancy resolution uses per-tenant db_name, but the fact base does not map each table to master or tenant DB.	Product owner	TBD
Is `conversations`/`chat_messages` duplicated by `inbox_messages`, and which is authoritative?	Overlapping inbox-related tables appear in the fact base with no ownership stated.	Sponsor + tech lead	TBD
How are `db_pass` values encrypted and where is the key held?	`tenants.db_pass` is stored, and encryptDbPass/decryptDbPass exist, but key management is not documented.	Sponsor + tech lead	TBD
Is `auto_replies.regex` and similar free-text field validated before use?	The fact base shows a regex trigger type but no validation rule.	Sponsor + tech lead	TBD
What is the source PHP version and which extensions are required?	phpVersion is empty in the fact base, which blocks build and behaviour parity planning.	Platform / ops	Wave 1
How will the 64 dynamic includes be resolved and validated?	Computed include paths are a major risk and their target class mapping is not in the fact base.	Sponsor + tech lead	Wave 1
Is the per-tenant database model retained in the Java target?	The redesign choice for bootstrapTenant is hibernate-multitenancy, but schema isolation and migration of tenant DBs are not specified.	Product owner	TBD
Are the cron-to-queue redesigns approved, and what are the delivery guarantees required?	Campaign and reminder sending must preserve idempotency and rate limiting (usleep loops) in the new design.	Tech lead	TBD
Should the setup and diagnostic tools (diag.php, fix.php) be removed or retained behind controls?	These tools are exposed routes and represent a security exposure if carried over.	Sponsor + tech lead	TBD
Which Java web framework and which message broker or job runtime should be standardised? The plan mixes queue, job-queue, message-queue, delayed-jobs, batch and JobRunr.	A single async backbone reduces operational surface. The fact base names patterns but not a concrete broker.	Tech lead	Planning

Question	Why it matters	Who can answer	Blocks which wave
Should the single-tenant cron variants be retired in favour of the SaaS variants? These are send_campaign.php vs send_campaign_saas.php and process_flows.php vs process_flows_saas.php.	Duplicate execution paths double migration and test effort.	Product owner	Planning
Must mobile API v1 and v2 both be preserved, or can v1 be deprecated?	Both are routed today, and the answer drives API compatibility scope for the mobile app.	Sponsor + tech lead	Planning
Should diag.php, fix.php and deploy/caddy-ask.php be migrated, replaced by actuator/ops tooling, or dropped?	Exposed diagnostic and repair endpoints are a security risk. Caddy on-demand TLS 'ask' behaviour must be kept if custom domains rely on it.	Sponsor + tech lead	Planning
Which database engine and version is in use? Also, are inbox_messages and chat_messages both authoritative?	The schema (enum, tinyint, json) implies MySQL/MariaDB but this is not stated. The overlapping message tables affect data-model consolidation.	Sponsor + tech lead	Planning
Which LLM provider and model does AIEngine call, and what are the outbound rate and throughput constraints for WhatsApp campaign sends?	Provider choice and rate limits are needed to design the AI client and to replace the usleep-based throttling in the campaign senders.	Sponsor + tech lead	Planning
Which PHP version is the source runtime, given the phpVersion field is empty?	Language-level migration gaps and library mappings depend on the source version.	Platform / ops	Wave 1
Is the mobile API v1 kept running in parallel with v2 during cutover, and for how long?	Mobile clients depend on JWT and refresh-token behaviour that must stay stable.	Sponsor + tech lead	Planning
What is the target deployment for the queue, job runner and websocket components?	Redesign choices name these components but the infrastructure target is not stated.	Tech lead	Planning
Are the setup and diagnostic tools (diag.php, fix.php, scripts) to be migrated or retired?	They contain dynamic includes and are reachable routes that affect scope and security.	Sponsor + tech lead	Planning
What is the PHP runtime version in use?	phpVersion is empty in the fact base and blocks compatibility assessment.	Sponsor + tech lead	TBD

Question	Why it matters	Who can answer	Blocks which wave
Are diag.php and fix.php reachable in production?	Both are routed and carry setup/diagnostic logic; exposure is not confirmed.	Sponsor + tech lead	TBD
Which of the 64 dynamic includes receive user-controlled input?	Computed include paths can be a code-execution risk; the fact base lists locations but not inputs.	Sponsor + tech lead	Wave 1
Does the capture's 164 distinct routes (660 fixtures) cover the production-critical paths, or should access-log intake be imported to refine priorities?	Fixture routeKeys now populate routeUsage; production access-log intake was not imported, so low-traffic or uncaptured routes may still need stakeholder confirmation.	Sponsor + tech lead	TBD
What is the PHP version to target for parity runs?	phpVersion is blank in the fact base.	Platform / ops	Wave 1
Does an existing automated test suite exist beyond the captured fixtures?	The fact base records fixtures and risks but no test suite or coverage data.	Sponsor + tech lead	TBD
Are there any open gaps that block parity sign-off?	openGaps is empty; it is unclear whether this means no gaps or an incomplete analysis.	Sponsor + tech lead	TBD
Are there stored procedures, triggers or views in the tenant or master schemas?	databaseLogic is empty in the fact base, so non-table database behaviour cannot be scoped.	Product owner	Wave 1
What is the MySQL/MariaDB version and character set of the existing tenant databases?	Required for correct JSON, decimal and utf8mb4 mapping; not captured in the fact base (phpVersion is also empty).	Platform / ops	Wave 1
Is a schema migration tool (Flyway/Liquibase) already in use, or are schema changes applied by PHP scripts such as scripts/apply_mobile_api_patch.php?	Determines how the baseline schema is captured and versioned for the Java target.	DBA / backend lead	Wave 1
Which tables hold personal data (leads phone/email, users email, tenants owner_email) and what retention applies?	Needed to plan encryption, masking and deletion handling during migration.	Product owner	Wave 1
What PHP version is the source system running?	phpVersion is empty in the fact base, which blocks migration baseline and compatibility analysis.	Platform / ops	Wave 1

Question	Why it matters	Who can answer	Blocks which wave
Do database logic, intake jobs and encoded files truly contain no items?	These collections are empty in the fact base and may reflect a capture/intake gap. Route usage is no longer empty — it is derived from capture fixture routeKeys (164 distinct routes).	Sponsor + tech lead	TBD
How will the 64 dynamic includes be resolved in Java?	Computed include paths cannot be translated directly and affect routing and module boundaries.	Sponsor + tech lead	Wave 1
Should diag.php, fix.php and the patch/admin scripts be retained, removed or restricted in production?	These setup and diagnostic tools expose operational risk and need an explicit disposition.	Sponsor + tech lead	TBD
How should the 3 usleep/sleep calls inside loops be replaced?	Blocking sleeps in campaign sending and auto-reply paths affect throughput and the planned queue redesign.	Sponsor + tech lead	TBD
What are the acceptance criteria for the proposed redesigns (queue, delayed-jobs, jobrunr, websocket)?	Redesign choices are listed without success measures, so parity cannot be verified.	Tech lead	TBD
What is the full role and permission matrix for tenant users and SaaS operators?	Only auth helper functions and role-related routes are captured, not the rules they enforce.	Product owner	TBD
Who owns validation of the 660 recorded fixtures against the Java target?	Fixtures are the main behavioural baseline for high-complexity units such as WhatsAppAPI and processEvent.	Sponsor + tech lead	TBD
Which PHP version is currently running in production?	The fact base leaves phpVersion empty, which affects the Java migration tooling and compatibility checks.	Platform / ops	Wave 1
Which tenant isolation model is in production: per-tenant databases or a shared schema?	Tenant bootstrap and DB connection logic will be redesigned differently for each model.	Product owner	TBD
Are the diagnostic and fix scripts (diag.php, fix.php) still reachable in production, and should they be retired?	These scripts carry high-complexity logic and dynamic includes and expose operational risk.	Sponsor + tech lead	TBD

13. What happens next

Wave plan

- Approved plan waves (corrected fact base): Wave 1 — 145 units (4713 LOC, 48 routes); Wave 2 — 98 units (7870 LOC, 35 routes); Wave 3 — 93 units (7134 LOC, 26 routes); Wave 4 — 146 units (5974 LOC, 25 routes); Wave 5 — 76 units (3758 LOC, 5 routes); Wave 6 — 35 units (1366 LOC). Total 593 units across six waves.
  Scope: WA Blast Pro, MEDIUM size band, 26436 PHP lines, plain PHP framework, 12 modules (common, users, inbox, ai, leads, campaigns, flows, appointments, rewards, billing, saas, mobileapi). The plan wave allocation is empty, so sequencing below is a draft that needs confirmation.

Approach

- Map each module to a Java package (com.wablastpro.<module>) and migrate module by module, starting with common since it carries shared infrastructure.
- Replace the per-request tenant bootstrap with a multitenancy approach (hibernate-multitenancy).
- Move asynchronous and scheduled work to queues and job runners instead of PHP cron scripts and inline loops.
- Resolve dynamic includes and in-loop sleeps before porting the affected units.

Redesign decisions already recorded

- bootstrapTenant: hibernate-multitenancy.
- processAutoReply: queue.
- processDueFlowEnrollments: delayed-jobs.
- send_campaign.php: job-queue; send_campaign_saas.php: message-queue.
- send_appointment_reminders.php: delayed-jobs; send_birthday_wishes.php: batch.
- run_tier_checks.php and billing_check.php: event-driven; process_flows_saas.php: jobrunr.
- inbox/poll.php: websocket.

Risk hotspots to plan for

- Dynamic include risk count is 64, concentrated in api/mobile, cron, includes and rewards; each needs a static replacement or explicit mapping.
- Three sleep-in-loop occurrences (cron/send_campaign.php, cron/send_campaign_saas.php, includes/webhook_processor.php) must be replaced by scheduled or queued delays.
- Highest complexity units: `WhatsAppAPI` (189), inbox/index.php (192, 1028 LOC), processEvent (134), handleBookingInbound (81), `AIEngine` (90). These should be migrated later with dedicated test coverage.

Proposed sequencing (draft)

1. common (config, tenant, session, SQL safety helpers).

2. users and saas (auth and tenant provisioning).
3. billing and rewards.
4. leads, campaigns and flows (queue and job redesign).
5. inbox, ai and appointments (webhook-heavy paths).
6. mobileapi (JWT and idempotent send endpoints).

Validation relies on the 660 recorded fixtures and the 60+ capture journeys, which should be used as regression baselines.

Indicative migration price range

Size band **medium** (26,436 PHP lines).

- Assessment (INR): ₹250000–350000
- Assessment (USD): $3000–4200
- Migration (INR): ₹687440–846080
- Migration (USD): $8196–10047
- Per 1,000 PHP lines (INR): ₹20000–32000
- Per 1,000 PHP lines (USD): $240–380

Ranges are published list bands, not a quote.

What MigrateForge needs from the client

- Named sponsor and technical contact.
- Pinned source commit and schema (or anonymised dump).
- Test accounts for capture and parity.
- Answers to the open questions that block wave 1.

Assessment credit

The Assessment fee is credited toward a migration of the same application that starts within **30 days** of this pack.