# Comprehensive Analysis & Migration Plan for the AiM Platform

## 1) Executive Summary

The current repository (ioqwfoihas, titled **Circle**) is a **Next.js** application inspired by the Linear project‑management tool. It tracks issues, projects and teams with a modern, responsive UI. The README describes the project as a management interface built with Next.js and shadcn/ui[[1]](https://github.com/rymleon23/ioqwfoihas/blob/master/README.md#L11-L18), and package.json reveals a client‑only stack with dependencies such as Next.js 15, React 19, Tailwind CSS, Radix UI components, Zustand, Recharts and Zod[[2]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L34-L57). There is no database or API layer, and data appears to live in local state.

AiM aims to transform this codebase into an **AI‑powered content platform** for **Creators** and **Brands**. The future system must support role‑based dashboards, campaign planning, AI‑assisted content creation, scheduling, asset management, analytics and administrative controls. This report analyses the existing repo, maps its capabilities to AiM requirements, identifies gaps and risks, proposes a target architecture and data model, and lays out a comprehensive migration roadmap with over **150 actionable tasks**.

## 2) Repo Inventory & Tech Stack

### High‑Level Tree (approximate)

The repository is a single Next.js app. Based on available files and typical Linear clone structures, the following tree (depth ≤3) reflects key components. Some paths are inferred because full directory listing is unavailable:

ioqwfoihas/  
├─ README.md # project overview[1]  
├─ package.json # dependencies and scripts[3]  
├─ tsconfig.json # TypeScript configuration  
├─ next.config.ts # minimal Next.js configuration  
├─ app/  
│ ├─ page.tsx # root page (redirects)  
│ ├─ layout.tsx # global layout (inferred)  
│ ├─ lndev-ui/ # main UI group for projects/boards  
│ │ ├─ team/[team]/[view]/ # dynamic routes for team boards & lists  
│ │ │ ├─ page.tsx # board or list view  
│ │ │ └─ layout.tsx # nested layout  
│ │ ├─ settings/… # settings pages (inferred)  
│ │ └─ …  
│ └─ (auth)/ # potential auth group (not implemented)  
├─ components/  
│ ├─ ui/ # shadcn wrappers  
│ ├─ board/ # Kanban components  
│ ├─ modal/ # dialogs and forms  
│ ├─ chart/ # analytics charts (recharts)  
│ └─ navigation/ # sidebars and menus  
├─ lib/  
│ ├─ store/ # Zustand stores for client state  
│ ├─ hooks/ # custom hooks (debounce, modal)  
│ └─ utils.ts # helper functions  
├─ public/ # static assets  
├─ styles/ # global CSS & Tailwind config  
├─ .husky/ # pre‑commit hooks  
└─ .eslintrc, .prettierrc # linting & formatting rules

### Tech & Versions

| Area | Details & Evidence |
| --- | --- |
| **Framework** | Next.js 15.2.4 – the app uses the App Router and React Server Components, as seen in the next dependency[[4]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L40-L42). |
| **Language** | TypeScript with strict checks (tsconfig enables strict mode). |
| **UI Library** | shadcn/ui built on Radix UI; numerous @radix-ui/react-\* dependencies[[5]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L14-L34). |
| **Styling** | Tailwind CSS v4 and tailwindcss-animate[[6]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L51-L54). |
| **State Mgmt** | Zustand for client state[[7]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L55-L57). |
| **Forms & Validation** | React Hook Form and Zod[[8]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L47-L57). |
| **Data Visualisation** | Recharts for charts[[9]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L50-L51). |
| **Utilities** | clsx, date-fns, uuid[[10]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L34-L56). |
| **Testing** | None present – no Jest, Vitest or Cypress. |
| **CI & Scripts** | Husky, lint-staged, ESLint and Prettier ensure basic code quality[[11]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L60-L71). |
| **Database/ORM** | Not present – there is no Prisma or other ORM. |

## 3) Architecture Assessment

1. **Single-app structure:** The codebase is a classic Next.js application without a monorepo. There are no workspaces or multiple packages.
2. **Client‑only implementation:** No API routes or server-side logic exist. Data is stored in client state (Zustand) and resets on refresh. This is unsuitable for AiM, which requires persistent data.
3. **UI & UX:** The app features a responsive layout with dark/light themes, sidebars and modals. The UI uses shadcn/ui components and Radix primitives. It includes features like command palette (cmdk) and charts (recharts) that can be repurposed.
4. **Routing:** The app uses Next.js App Router. The root page redirects to a team board, indicating a board/list view structure. Nested dynamic routes manage team and view parameters.
5. **State Management:** Zustand stores handle local data. Without a backend, this is appropriate; however, AiM requires React Query or SWR to fetch server data.
6. **Authentication:** There is no authentication module. All pages are publicly accessible; there is no concept of roles or users.
7. **Configuration & Environment:** Minimal configuration exists – next.config.ts only disables dev indicators; there is no .env.example or environment variables. Secret management and environment segregation are missing.
8. **Quality & Tooling:** Basic linting and formatting are in place, but there are no tests or continuous integration. Developer experience will degrade as complexity grows.
9. **Security:** Without auth and input validation, the current app has no security boundaries. All actions are performed client-side, which is insecure for sensitive content and analytics.

## 4) AiM Mapping (Reuse / Extend / Replace / Remove)

| Existing Module/Screen | AiM Feature Mapping | Action | Rationale |
| --- | --- | --- | --- |
| **UI Shell (layout, navigation)** | Base dashboard shell for all roles | **Reuse & Extend** | Retain responsive design and theme toggles; extend navigation to include AiM modules (Campaigns, Content Studio, Calendar, Assets, Analytics, Settings). |
| **Team/Project Boards** | Campaign Kanban boards | **Extend** | Convert issue cards to campaign cards with statuses (Draft, Active, Completed). Add drag‑and‑drop to reorder campaigns and update status persistently. |
| **List/Issue Views** | Content lists within campaigns | **Replace** | Replace issue list with content items; integrate AI status (Draft, AI‑generated, Approved). |
| **Zustand Stores & Mock Data** | Server data via React Query | **Replace** | Remove mock stores; use zustand only for UI state while data persists in Postgres via API. |
| **Modals & Forms** | Campaign & content creation wizards | **Reuse & Extend** | Reuse modal patterns; add multi‑step wizards, form validation (zod) and AI integration for content suggestions. |
| **Charts (Recharts)** | Campaign & content analytics | **Reuse & Extend** | Use existing charts for metrics like impressions, click‑through and ROI; connect to analytics API. |
| **Command palette** | AI assistant & global search | **Extend** | Augment the palette to accept AI prompts and execute actions (e.g. “Generate LinkedIn post for campaign X”). |
| **No Auth & RBAC** | Authentication & role‑based access | **Add** | Introduce NextAuth and Prisma for user management; roles: Creator, Brand, Admin (and optional Reviewer/Publisher). |
| **No API & ORM** | API endpoints & database with Prisma | **Add** | Create API routes/tRPC for campaigns, content, schedules, assets and analytics; use Prisma with Postgres for storage. |
| **No Tests or CI** | Testing framework & CI pipelines | **Add** | Introduce unit, integration and e2e tests; configure GitHub Actions for lint, test, build and health checks. |
| **No Asset management** | Asset storage (images/videos) | **Add** | Integrate file upload to S3 or local storage; create Asset library UI and API. |
| **No analytics/event tracking** | Analytics events & dashboards | **Add** | Record views/clicks; compute and display metrics across campaigns and content. |
| **No AI integration** | AI‑powered content generation & assistance | **Add** | Add server route to call AI (e.g. OpenAI API) and UI components to generate, refine and translate content. |
| **No i18n** | Multilingual support (English/Vietnamese) | **Extend** | Provide translations for UI strings; integrate next-intl. |
| **No settings module** | Settings for profile, org, billing, AI preferences | **Add** | Build pages to manage user/organization settings, API keys and preferences. |

## 5) Gaps, Risks & Mitigations

### Gaps

1. **Data persistence:** Without a database, information resets on refresh. **Mitigation:** Introduce Postgres via Prisma; define schema for campaigns, content, assets, schedules, analytics and organizations.
2. **Authentication & RBAC:** No login or role management. **Mitigation:** Use NextAuth.js with credential/OAuth providers; implement role enumeration and RBAC middleware.
3. **API layer:** CRUD operations are missing. **Mitigation:** Create Next.js API routes or tRPC routers with proper validation (zod) and caching.
4. **Testing & CI:** There are no automated tests or CI pipelines. **Mitigation:** Add Jest/RTL or Vitest for unit tests, Playwright for e2e tests, and GitHub Actions for continuous integration.
5. **Security:** Lacks input sanitisation, rate limiting, and secure secrets management. **Mitigation:** Validate and sanitise inputs server side; store secrets in environment variables; implement throttling and secure cookie settings.
6. **Observability:** No logging or error tracking. **Mitigation:** Use logging (pino/winston) and integrate error monitoring (Sentry). Record analytics events for performance and feature adoption.
7. **Internationalisation:** UI is likely English-only. **Mitigation:** Add next-intl and translation files for required languages; detect locale from user profile or browser.
8. **Scalability:** Single repository may become monolithic. **Mitigation:** Use modular architecture with clear boundaries; consider splitting into separate packages later (e.g. @aim/core, @aim/ui).

### Risks & Mitigations

| Risk | Impact | Likelihood | Mitigation |
| --- | --- | --- | --- |
| **Migration complexity** | Large refactor may introduce bugs | High | Plan incremental PRs, maintain working state, write automated tests. |
| **Unstable dependencies** | Next.js 15 is still evolving | Medium | Pin versions; monitor releases; test upgrades before merging. |
| **Security vulnerabilities** | New auth and file uploads could be exploited | Medium | Use secure libs; follow OWASP guidelines; implement rate limiting and file type validation. |
| **Performance bottlenecks** | Campaign board may have many items | Medium | Implement pagination/virtualisation; lazy load modules; use React Query caching. |
| **Team adoption & training** | New stack (Prisma, NextAuth) requires learning | Medium | Provide documentation and code samples; organise workshops. |
| **Regulatory compliance (e.g. GDPR)** | Handling user data incorrectly could cause fines | Low | Ensure privacy by design; implement data retention policies; add cookie consent. |

## 6) Target Architecture for AiM

### High‑Level Architecture Diagram

The following Mermaid diagram illustrates the proposed architecture for AiM, introducing a back‑end and supporting services:

flowchart TD  
 subgraph Client  
 U[Creators/Brands/Admins] -->|Browser| FE[Next.js 15 App]  
 FE -->|Fetch| API  
 FE -->|Auth Hooks| AuthClient  
 end  
 subgraph Server  
 API --> DB[(PostgreSQL via Prisma)]  
 API --> AIService[(External AI API)]  
 API --> Storage[(Asset Storage: S3/UploadThing)]  
 API --> Analytics[(Analytics Service)]  
 API --> RBAC[(RBAC Middleware)]  
 AuthService[(NextAuth.js)] --> DB  
 RBAC --> AuthService  
 end  
 Analytics --> DB  
 FE -.->|WebSockets| Notifications[(Real‑time Notifications)]

**Description:** \* **Frontend (Next.js)** renders role‑specific pages, manages UI state with Zustand and fetches data via React Query from the API. \* **API Layer** (Next.js API routes or tRPC) exposes endpoints for campaigns, content, assets, schedules, analytics and AI integration. Zod validation and RBAC middleware protect these routes. \* **Authentication Service** uses NextAuth.js for credential/OAuth login; sessions stored in Postgres. \* **Database (PostgreSQL)** stores all persistent data with a normalized schema; Prisma manages migrations and type‑safe queries. \* **AI Service** proxies calls to external AI provider(s) for content generation, summarisation, translation and ideation. \* **Asset Storage** handles file uploads (images, videos) to S3 or local storage; metadata recorded in DB. \* **Analytics Service** collects event data from clients and provides aggregated metrics; data persisted in DB for reporting. \* **Real‑time Notifications** (optional) deliver updates to clients via WebSockets or Pusher (e.g. content approval status).

### Proposed Data Model

| Entity | Key Fields & Notes | Relationships |
| --- | --- | --- |
| **User** | id (UUID), name, email (unique), passwordHash, role (enum: creator, brand, admin, reviewer, publisher), organizationId, timestamps | Many users belong to one **Organization**; users create **Campaigns**, **Contents**, **Schedules**, **Assets**, **AnalyticsEvents**. |
| **Organization** | id, name, slug, createdAt, updatedAt | Has many **Users**, **Campaigns**, **Assets**. |
| **Campaign** | id, organizationId, createdById (User), name, description, status (draft/active/completed), budget, startDate, endDate, createdAt, updatedAt | Belongs to an **Organization** and a **Creator**; has many **Contents** and **Schedules**; can have many **AnalyticsEvents** via its contents. |
| **Content** | id, campaignId, authorId, assetId (optional), title, body, type (blog/article/video), aiGenerated (bool), status (draft/submitted/approved/published/rejected), metadata (JSON), timestamps | Belongs to a **Campaign**; optionally references an **Asset**; has many **Schedules**; has many **AnalyticsEvents**. |
| **Asset** | id, organizationId, uploadedById (User), fileName, fileType, size, url, createdAt, updatedAt | Belongs to an **Organization**; can be linked to multiple **Contents**. |
| **Schedule** | id, contentId, campaignId, scheduledAt (DateTime), platform (enum: blog, linkedin, facebook, youtube, tiktok, etc.), status (scheduled/published/cancelled), createdById | Links **Content** to a specific publication time on a platform; belongs to **Campaign**. |
| **AnalyticsEvent** | id, contentId, userId (viewer or actor), eventType (view/click/like/share/conversion), timestamp, metadata (JSON) | Logs interactions for analytics; aggregated for metrics. |
| **Role** (optional table) | id, name, permissions (JSON or relation) | Many‑to‑many with **User** if more granular RBAC is needed. |

This schema sets the foundation for AiM. Additional tables (Tags, Categories, Comments, Webhooks) can be added later.

### UI Shell & Role‑Based Navigation

The existing layout will be enhanced to provide clear navigation per role:

* **Creator**: Dashboard with quick access to campaigns, Content Studio, Calendar, Assets and Analytics. Includes AI assistant for ideation and content generation. Shows assignments and deadlines.
* **Brand**: Dashboard summarising campaigns across creators, approval queues, budget and ROI analytics. Enables content approval, asset review and schedule oversight.
* **Admin**: Portal for managing organizations, users, roles and platform settings. Provides visibility across all data and ability to configure global settings (billing, AI quotas).

## 7) Delta Plan (File‑Level)

The delta plan enumerates file additions, modifications and removals necessary to evolve the current repo into AiM. File paths may vary in the actual repository; adjust accordingly.

| Action | File/Path | Description |
| --- | --- | --- |
| **ADD** | prisma/schema.prisma | Define models for User, Organization, Campaign, Content, Asset, Schedule, AnalyticsEvent and optional Role. |
| **ADD** | prisma/migrations/\* | Auto‑generated migrations for schema changes. |
| **ADD** | lib/prisma.ts | Singleton Prisma client with error handling. |
| **ADD** | app/api/auth/[...nextauth]/route.ts | NextAuth configuration with credentials and optional OAuth providers. |
| **ADD** | app/api/auth/register/route.ts | Endpoint to register new users and organizations; hashes passwords and returns session token. |
| **ADD** | app/api/campaigns/route.ts | REST or tRPC handler for listing and creating campaigns. |
| **ADD** | app/api/campaigns/[id]/route.ts | Handler for reading, updating and deleting a campaign by ID. |
| **ADD** | app/api/contents/route.ts | Endpoint for listing and creating content; filters by campaignId. |
| **ADD** | app/api/contents/[id]/route.ts | Handler for reading, updating and deleting a content item. |
| **ADD** | app/api/assets/route.ts | Handler for uploading assets (multipart/form-data), listing assets and deleting assets. |
| **ADD** | app/api/schedules/route.ts | Endpoint to create and list schedules. |
| **ADD** | app/api/schedules/[id]/route.ts | Handler to update or cancel a schedule. |
| **ADD** | app/api/analytics/route.ts | Endpoint to record analytics events and fetch aggregated metrics. |
| **ADD** | app/api/ai/generate/route.ts | Proxy to AI service for content generation; caches responses. |
| **ADD** | app/middleware.ts | Next.js middleware for session and RBAC; redirects unauthenticated users. |
| **ADD** | lib/rbac.ts | Helpers to check permissions based on user role. |
| **ADD** | lib/hooks/useCurrentUser.ts | Custom hook to fetch current user data via /api/auth/session. |
| **ADD** | app/(auth)/login/page.tsx | Login page using shadcn form components; calls NextAuth signIn. |
| **ADD** | app/(auth)/register/page.tsx | Registration page with zod validation; calls register API. |
| **ADD** | app/(dashboard)/creator/layout.tsx | Creator dashboard layout with sidebar and topbar. |
| **ADD** | app/(dashboard)/creator/page.tsx | Creator dashboard home showing campaign summary and quick actions. |
| **ADD** | app/(dashboard)/brand/layout.tsx | Brand dashboard layout. |
| **ADD** | app/(dashboard)/brand/page.tsx | Brand dashboard page summarising campaigns and metrics. |
| **ADD** | app/(dashboard)/admin/page.tsx | Admin dashboard for user/org management. |
| **ADD** | app/campaigns/page.tsx | Campaign list view with table and actions. |
| **ADD** | app/campaigns/board/page.tsx | Campaign Kanban board; reused and extended from existing board component. |
| **ADD** | app/campaigns/new/page.tsx | Multi‑step wizard for creating a new campaign. |
| **ADD** | app/campaigns/[id]/page.tsx | Campaign details page; shows overview, content, schedule and analytics. |
| **ADD** | app/content/[id]/page.tsx | Content editor page with AI assistant and preview. |
| **ADD** | app/content/new/page.tsx | Page to create new content from scratch or AI suggestions. |
| **ADD** | app/calendar/page.tsx | Calendar view for scheduled posts; uses react-day-picker. |
| **ADD** | app/assets/page.tsx | Asset library UI; lists uploaded files and provides upload. |
| **ADD** | app/settings/page.tsx | User and organization settings page (profile, password, AI preferences). |
| **ADD** | components/forms/CampaignForm.tsx | Form component for campaign creation/edit using react-hook-form and zod. |
| **ADD** | components/forms/ContentForm.tsx | Form for content; includes AI panel and asset picker. |
| **ADD** | components/ai/AssistantPanel.tsx | Component to display AI suggestions, prompts and generation progress. |
| **ADD** | components/dashboard/SummaryCard.tsx | Card component for dashboard metrics. |
| **ADD** | components/dashboard/CampaignList.tsx | List component for campaigns. |
| **ADD** | components/dashboard/BrandMetrics.tsx | Displays aggregated brand metrics. |
| **ADD** | components/navigation/Sidebar.tsx | Sidebar that renders different menus based on user role. |
| **ADD** | components/navigation/TopBar.tsx | Top bar with search, notifications and user menu. |
| **ADD** | components/navigation/MenuItem.tsx | Menu item component used in sidebar. |
| **ADD** | .env.example | Template for environment variables (DATABASE\_URL, NEXTAUTH\_SECRET, S3 keys, AI API keys). |
| **ADD** | .github/workflows/ci.yml | GitHub Actions workflow: install deps, lint, type‑check, run tests, build and run a health check on /api/health. |
| **ADD** | .github/workflows/e2e.yml | Workflow to run Playwright e2e tests on PRs. |
| **ADD** | Dockerfile & docker-compose.yml | Containerize the app and run Postgres locally. |
| **ADD** | scripts/publish-worker.ts | CRON worker that publishes scheduled content at the scheduled time. |
| **MODIFY** | package.json | Add new dependencies (prisma, @prisma/client, next-auth, bcrypt, zod, @uploadthing/client, @tanstack/react-query, next-intl, jest, @testing-library/react, playwright) and scripts (test, prisma generate, seed). |
| **MODIFY** | tsconfig.json | Update paths for new folders (@/prisma, @/components/forms, etc.) and enable strict property initialization. |
| **MODIFY** | next.config.ts | Configure i18n, image domains, optional server actions and experimental features. |
| **MODIFY** | app/layout.tsx | Wrap with <SessionProvider> and <QueryClientProvider>; include translations and theme provider. |
| **MODIFY** | app/page.tsx | Replace redirect with logic to route user to appropriate dashboard based on role. |
| **MODIFY** | components/board | Rename to handle campaign statuses; persist state changes via API calls. |
| **MODIFY** | components/navigation | Update sidebar/menu to reflect AiM modules and role access. |
| **MODIFY** | styles/globals.css | Introduce AiM brand colours, fonts and typography. |
| **REMOVE** | Mock data and local JSON fixtures in lib/store | Data will be served from DB; remove local store or convert to UI state only. |
| **REMOVE** | Deprecated issue/project routes under lndev-ui | Replace with new campaigns/content routes. |
| **REMOVE** | Redundant assets and placeholder images | Use real assets via the Asset library; remove unused icons. |

## 8) Roadmap – 30/60/90 Day Plan

### **0–30 days: Foundation Setup**

1. **Database & ORM (P0):**
2. Install and configure Prisma and Postgres locally. Create initial schema and run migrations.
3. Add lib/prisma.ts client.
4. Write seeding script for admin user and sample org.
5. **Authentication & RBAC (P0):**
6. Integrate NextAuth with credential provider; implement login and registration APIs.
7. Create User and Organization models; assign roles (Creator, Brand, Admin) on registration.
8. Add middleware for route protection.
9. **Initial API endpoints (P0–P1):**
10. Scaffold CRUD endpoints for Campaigns and Content.
11. Implement zod validation and RBAC checks.
12. **Dashboard shell (P0):**
13. Build role‑aware sidebar and topbar.
14. Create minimal Creator and Brand dashboard pages showing lists of campaigns.
15. **CI/CD & Dev Experience (P0):**
16. Add .env.example, GitHub Actions for lint and type checking.
17. Configure Prettier, ESLint and Husky.
18. **Docs & Onboarding (P0–P1):**
19. Update README with new stack and dev setup.
20. Document environment setup and migration commands.

### **31–60 days: Core Features**

1. **Campaign Management (P0–P1):**
2. Implement Kanban board for campaigns with drag‑and‑drop to update status.
3. Build multi‑step Campaign creation wizard.
4. Add table view of campaigns with sorting/filtering.
5. **Content Studio (P0–P2):**
6. Develop rich text editor with AI assistant panel.
7. Integrate AI generation endpoint; allow saving drafts and attaching assets.
8. Implement content list inside campaign details page.
9. **Calendar & Scheduling (P1):**
10. Implement calendar view; show scheduled posts and allow scheduling new posts.
11. Add schedules API; implement cron worker to publish content.
12. **Asset Library (P1):**
13. Integrate file upload (S3/UploadThing); build asset management UI.
14. Connect assets to content items.
15. **Analytics & Reporting (P2):**
16. Record analytics events via API; create basic dashboard graphs.
17. Display campaign performance metrics in Brand dashboard.
18. **Internationalisation (P2):**
19. Integrate next-intl; add English and Vietnamese translations; implement language switcher in settings.
20. **Testing & Quality (P1–P2):**
21. Add unit tests for utils/components, integration tests for API routes, and e2e tests with Playwright (login, create campaign, publish content).

### **61–90 days: Advanced & Polishing**

1. **Advanced RBAC (P2):**
2. Introduce Reviewer and Publisher roles; implement content approval flows.
3. Add approval queue in Brand dashboard.
4. **Notifications & Real‑time (P3):**
5. Implement real‑time notifications for approvals, publishing status and comments using WebSockets or Pusher.
6. **Observability & Monitoring (P2):**
7. Add structured logging (pino/winston), integrate error tracking (Sentry) and metrics collection.
8. Provide health check endpoint for CI.
9. **Performance Optimisation (P2–P3):**
10. Implement pagination/virtualisation for long lists; optimise images; enable caching strategies.
11. Use SSG/ISR for content pages where appropriate.
12. **Collaboration Features (P3):**
13. Add commenting on content with mentions; implement notifications.
14. Provide version history and change tracking for content.
15. **Deployment & Scalability (P2–P3):**
16. Create Dockerfile and docker-compose for local and production deployment.
17. Deploy to Vercel or AWS; implement environment-specific configs.
18. **Extensibility (P3):**
19. Define plugin architecture for analytics widgets and AI models.
20. Document extension points for partners.

## 9) Backlog – Detailed To‑Do (150+ Tasks)

The backlog is grouped by epic with stories and granular tasks. Each task includes acceptance criteria, files touched, effort estimate (Small/Medium/Large) and priority (P0–P3). This provides a ready‑to‑use checklist for GitHub Issues.

### Epic: Authentication & RBAC

#### Story: Install and Configure NextAuth.js

* [ ] **Install Auth dependencies:** Add next-auth, @types/next-auth, bcrypt, and jsonwebtoken to package.json; run pnpm install.  
  *Acceptance:* Dependencies appear in package.json; pnpm install completes without errors.  
  *Files:* package.json, pnpm-lock.yaml.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Create NextAuth route:** Implement app/api/auth/[...nextauth]/route.ts with credential provider using email & password. Define session and JWT callbacks.  
  *Acceptance:* Users can log in via /api/auth/callback/credentials; tokens are returned.  
  *Files:* app/api/auth/[...nextauth]/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Hash passwords:** Write helper lib/auth.ts with hashPassword() and verifyPassword() using bcrypt.  
  *Acceptance:* Passwords hashed correctly; verification works in tests.  
  *Files:* lib/auth.ts.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Create register endpoint:** In app/api/auth/register/route.ts, accept name, email, password, create user and organization, hash password, return session token.  
  *Acceptance:* New users can register; duplicate emails return 400; created user appears in DB.  
  *Files:* app/api/auth/register/route.ts, lib/prisma.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Add session provider:** Wrap app/layout.tsx with <SessionProvider>.  
  *Acceptance:* useSession() hook in components returns user data.  
  *Files:* app/layout.tsx.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Implement login page:** Create app/(auth)/login/page.tsx with form, zod validation and call signIn(). Show error messages on invalid credentials.  
  *Acceptance:* Users can log in; on success they are redirected to dashboard; invalid credentials show error.  
  *Files:* app/(auth)/login/page.tsx, components/forms/LoginForm.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Implement registration page:** Create app/(auth)/register/page.tsx with form to create account and organization; call register API.  
  *Acceptance:* Users can register; organisation is created; user becomes admin; redirected to dashboard.  
  *Files:* app/(auth)/register/page.tsx, components/forms/RegisterForm.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Setup logout:** Add logout button to user menu in top bar; call signOut() to clear session.  
  *Acceptance:* Clicking logout terminates session and redirects to login page.  
  *Files:* components/navigation/TopBar.tsx, lib/hooks/useCurrentUser.ts.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Session API route:** Expose /api/auth/session to return current user info; used by useCurrentUser hook.  
  *Acceptance:* Authenticated requests return user; unauthenticated return null.  
  *Files:* app/api/auth/session/route.ts.  
  *Effort:* S  
  *Priority:* P0

#### Story: Role Management & RBAC

* [ ] **Define role types:** Create lib/roles.ts enumerating roles (creator, brand, admin, reviewer, publisher).  
  *Acceptance:* Roles exported and used across app.  
  *Files:* lib/roles.ts.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Add Role field to User model:** Modify schema.prisma to include role enum field with default creator. Run migration.  
  *Acceptance:* DB schema updated; existing users default to creator.  
  *Files:* prisma/schema.prisma, migration.  
  *Effort:* S  
  *Priority:* P0
* [ ] **RBAC helper:** Implement lib/rbac.ts with functions hasRole(user, role) and hasPermission(user, permission).  
  *Acceptance:* Tests verify correct access decisions.  
  *Files:* lib/rbac.ts, tests/lib/rbac.test.ts.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Middleware for protected routes:** Create middleware.ts to check session and roles before allowing access to certain routes.  
  *Acceptance:* Unauthenticated requests to /dashboard redirect to /login; users without required role get 403.  
  *Files:* middleware.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Assign roles on registration:** Extend registration endpoint to set role based on query param or default; admin role assigned when first user of organisation.  
  *Acceptance:* Role stored in DB; new users can sign up as creators or brands; admin created by default.  
  *Files:* app/api/auth/register/route.ts.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Role switcher for testing:** In development only, provide a dropdown in settings to switch roles for ease of testing.  
  *Acceptance:* Changing role updates session and UI; only available in dev.  
  *Files:* app/settings/page.tsx.  
  *Effort:* S  
  *Priority:* P3
* [ ] **Role-based navigation:** Modify Sidebar.tsx to show/hide menu items based on user role (e.g. Campaigns for all, Approvals only for brands, Admin for admins).  
  *Acceptance:* Sidebar reflects correct navigation per role; hidden items inaccessible via URL.  
  *Files:* components/navigation/Sidebar.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Role-based API checks:** Ensure each API route calls RBAC helper to verify user has permission to perform action (e.g. only brand can approve content).  
  *Acceptance:* API returns 403 when unauthorized; tests cover scenarios.  
  *Files:* all API route files; lib/rbac.ts.  
  *Effort:* M  
  *Priority:* P0

### Epic: Database & Prisma

#### Story: Schema Definition

* [ ] **Initialize Prisma:** Run npx prisma init to create prisma/schema.prisma and .env.  
  *Acceptance:* prisma folder created; schema.prisma contains datasource pointing to DATABASE\_URL.  
  *Files:* prisma/schema.prisma, .env.example.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Define User & Organization models:** Add User and Organization models with necessary fields and relations.  
  *Acceptance:* prisma validate passes; migration applies.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Define Campaign model:** Create Campaign model with fields (id, name, description, status, budget, startDate, endDate, createdBy, organizationId).  
  *Acceptance:* Generated DB table matches model; relations defined.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Define Content model:** Add Content model with fields (id, title, body, type, aiGenerated, status, authorId, campaignId, assetId, metadata).  
  *Acceptance:* Migration creates Content table; relations to Campaign, User, Asset established.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Define Asset model:** Add Asset model with fields (id, fileName, fileType, url, size, uploadedById, organizationId).  
  *Acceptance:* Migration runs successfully; file references stored.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Define Schedule model:** Add Schedule model linking content to publication time and platform; include status.  
  *Acceptance:* Table created; relation to Content and Campaign validated.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Define AnalyticsEvent model:** Add model to log user interactions with content; include metadata JSON.  
  *Acceptance:* Table created; indexes on timestamp for fast querying.  
  *Files:* prisma/schema.prisma.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Define optional Role model:** If granular permissions needed, create Role model with name and permissions; create many‑to‑many relation with Users via UserRole junction.  
  *Acceptance:* Role and UserRole tables created; relation usable via Prisma client.  
  *Files:* prisma/schema.prisma.  
  *Effort:* M  
  *Priority:* P3

#### Story: Migrations & Seeding

* [ ] **Generate initial migration:** Run npx prisma migrate dev --name init after defining models.  
  *Acceptance:* Migration files appear in prisma/migrations/\*; DB schema updated.  
  *Files:* prisma/migrations/\*.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Write seed script:** Create scripts/seed.ts to insert default roles, admin user, example organisation, sample campaigns and content.  
  *Acceptance:* Running script populates DB; safe to rerun.  
  *Files:* scripts/seed.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Add seed command:** Update package.json script seed to run tsx scripts/seed.ts.  
  *Acceptance:* Running pnpm seed executes the seed script.  
  *Files:* package.json.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Document DB setup:** Add instructions in README to install Postgres, set DATABASE\_URL, run migrations and seed.  
  *Acceptance:* Developers can follow docs to set up DB.  
  *Files:* README.md.  
  *Effort:* S  
  *Priority:* P1

### Epic: API Layer

#### Story: Campaign Endpoints

* [ ] **List campaigns:** In app/api/campaigns/route.ts, implement GET handler to return campaigns for current user's organisation; support pagination (page, limit) and search (q).  
  *Acceptance:* GET returns paginated campaigns with total pages; search returns filtered results; unauthorized returns 403.  
  *Files:* app/api/campaigns/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Create campaign:** Implement POST handler; validate input via zod; assign createdBy to current user; set status draft.  
  *Acceptance:* Valid input creates campaign in DB and returns record; invalid input returns 400.  
  *Files:* app/api/campaigns/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Get campaign by ID:** Create GET handler in app/api/campaigns/[id]/route.ts; fetch campaign and related contents, schedules and analytics summary.  
  *Acceptance:* Returns campaign details with relations; 404 if not found; 403 if user not authorized.  
  *Files:* app/api/campaigns/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Update campaign:** Implement PUT handler to update name, description, dates, status and budget; use zod for input.  
  *Acceptance:* Changes persist; invalid fields return 400; unauthorized returns 403.  
  *Files:* app/api/campaigns/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Delete campaign:** Implement DELETE handler; cascade delete content, schedules and analytics or set them orphaned (per business rules).  
  *Acceptance:* Campaign removed; associated records handled accordingly; unauthorized returns 403.  
  *Files:* app/api/campaigns/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Campaign analytics summary:** Add GET /app/api/campaigns/[id]/analytics endpoint to return aggregated events (views, clicks, conversions).  
  *Acceptance:* Response includes metrics aggregated by date range; uses DB queries or materialised views.  
  *Files:* app/api/campaigns/[id]/analytics/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Campaign duplication:** Provide endpoint POST /campaigns/[id]/duplicate to clone a campaign including its content.  
  *Acceptance:* New campaign created with “Copy of …” name; content duplicated; schedules not copied.  
  *Files:* app/api/campaigns/[id]/duplicate/route.ts.  
  *Effort:* M  
  *Priority:* P3

#### Story: Content Endpoints

* [ ] **List content:** In app/api/contents/route.ts, implement GET handler to return content items for a campaign; support filters by status and type.  
  *Acceptance:* Response returns paginated list; unauthorized returns 403.  
  *Files:* app/api/contents/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Create content:** Implement POST handler to create new content; accept campaignId, title, body, type, assetId and aiGenerated.  
  *Acceptance:* Valid data creates content; returns created record; invalid returns 400.  
  *Files:* app/api/contents/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Get content by ID:** Add GET handler in app/api/contents/[id]/route.ts to fetch content with related asset, author and schedules.  
  *Acceptance:* Returns content details; 404 if not found; 403 if unauthorized.  
  *Files:* app/api/contents/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Update content:** Implement PUT handler to update title, body, asset, type and status; restrict editing based on role and status (e.g. approved content cannot be edited by creator).  
  *Acceptance:* Role and status checks enforced; updates persist; invalid input returns 400.  
  *Files:* app/api/contents/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Delete content:** Add DELETE handler to remove content; cascade delete schedules and analytics events.  
  *Acceptance:* Content removed; schedules and analytics handled; unauthorized returns 403.  
  *Files:* app/api/contents/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Submit content for approval:** Add endpoint POST /contents/[id]/submit to mark content as submitted; triggers notification to brand.  
  *Acceptance:* Content status changes to submitted; brand sees item in approval queue; unauthorized returns 403.  
  *Files:* app/api/contents/[id]/submit/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Approve or reject content:** Create endpoint POST /contents/[id]/approve and /reject accessible to brand role.  
  *Acceptance:* Status updates accordingly; author notified; reject includes optional feedback.  
  *Files:* app/api/contents/[id]/approve/route.ts, app/api/contents/[id]/reject/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **AI generation endpoint:** Implement app/api/ai/generate/route.ts using OpenAI API; accept prompt and options (tone, length); store generated text in DB for auditing.  
  *Acceptance:* Valid requests return generated text; caching implemented; errors handled gracefully.  
  *Files:* app/api/ai/generate/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Content translation endpoint:** Add POST /ai/translate to translate content into target language.  
  *Acceptance:* Returns translated text; logs translation usage.  
  *Files:* app/api/ai/translate/route.ts.  
  *Effort:* S  
  *Priority:* P3
* [ ] **Content summarisation endpoint:** Add POST /ai/summarise for summarising long content.  
  *Acceptance:* Returns summary; uses AI service; caches result.  
  *Files:* app/api/ai/summarise/route.ts.  
  *Effort:* S  
  *Priority:* P3

#### Story: Asset Endpoints

* [ ] **Upload asset:** In app/api/assets/route.ts, implement POST handler using UploadThing or AWS S3. Accept files, validate type and size, store to storage, record metadata in DB.  
  *Acceptance:* Uploading image or video returns URL and DB record; invalid file type returns 400.  
  *Files:* app/api/assets/route.ts, lib/storage.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **List assets:** Implement GET handler to return assets belonging to organisation; support pagination and search by filename.  
  *Acceptance:* Response returns list; unauthorized returns 403.  
  *Files:* app/api/assets/route.ts.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Delete asset:** Add DELETE handler at app/api/assets/[id]/route.ts; remove file from storage and record from DB if not linked to content.  
  *Acceptance:* Asset removed; protected if linked to content unless force flag.  
  *Files:* app/api/assets/[id]/route.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Generate signed upload URLs:** Provide endpoint to return pre‑signed S3 URLs for direct browser uploads (if using S3).  
  *Acceptance:* GET returns URL; client can upload directly; server records metadata after upload.  
  *Files:* app/api/assets/upload-url/route.ts.  
  *Effort:* M  
  *Priority:* P3

#### Story: Schedule Endpoints

* [ ] **List schedules:** Implement GET handler in app/api/schedules/route.ts to list schedules for campaign or content; filter by status and date range.  
  *Acceptance:* Returns schedule array; unauthorized returns 403.  
  *Files:* app/api/schedules/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Create schedule:** POST handler to schedule content to specific platform and time; validate date is in future; enforce that content is approved.  
  *Acceptance:* Creates schedule; returns new record; invalid input returns 400.  
  *Files:* app/api/schedules/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Update schedule:** PUT handler in app/api/schedules/[id]/route.ts to reschedule or change platform; restrict editing if already published.  
  *Acceptance:* Update persists; status remains consistent; unauthorized returns 403.  
  *Files:* app/api/schedules/[id]/route.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Cancel schedule:** DELETE or POST /cancel handler to cancel publication; mark status cancelled.  
  *Acceptance:* Schedule removed or flagged; content remains unpublished; notifications triggered.  
  *Files:* app/api/schedules/[id]/route.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Publish worker:** Create script scripts/publish-worker.ts that runs every minute to find schedules due for publication; update content status to published; record analytics event; send notifications.  
  *Acceptance:* Cron job publishes scheduled content at correct time; logs success and failures.  
  *Files:* scripts/publish-worker.ts, package.json (script).  
  *Effort:* M  
  *Priority:* P3

#### Story: Analytics Endpoints

* [ ] **Record analytics event:** In app/api/analytics/route.ts, implement POST handler to record events with contentId, eventType, userId (optional) and metadata.  
  *Acceptance:* Event saved in DB; invalid eventType returns 400; no duplication for idempotent events.  
  *Files:* app/api/analytics/route.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **List events:** Add GET handler to return events filtered by contentId, campaignId, eventType, dateRange.  
  *Acceptance:* Returns event list; unauthorized returns 403.  
  *Files:* app/api/analytics/route.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Aggregate metrics:** Implement endpoint /api/analytics/summary that aggregates events by campaign or content; compute counts per day and conversion rates.  
  *Acceptance:* Response provides aggregated data; performance acceptable with indexes.  
  *Files:* app/api/analytics/summary/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Export analytics:** Provide endpoint /api/analytics/export to export metrics to CSV/Excel; restrict to admin or brand.  
  *Acceptance:* Download file contains data; large exports streamed to avoid memory issues.  
  *Files:* app/api/analytics/export/route.ts.  
  *Effort:* M  
  *Priority:* P3

### Epic: Frontend – Global Layout & Navigation

#### Story: Providers & Setup

* [ ] **React Query provider:** Install @tanstack/react-query; create lib/queryClient.ts and wrap root layout with <QueryClientProvider> and <Hydrate> for SSR.  
  *Acceptance:* Components can call useQuery and useMutation; no hydration errors.  
  *Files:* package.json, lib/queryClient.ts, app/layout.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Internationalisation provider:** Install next-intl; configure i18n in next.config.ts; create locales folder with English and Vietnamese translations; wrap layout with <IntlProvider>.  
  *Acceptance:* UI strings translate based on locale; fallback to English; user can change language in settings.  
  *Files:* package.json, next.config.ts, locales/en.json, locales/vi.json, app/layout.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Theme provider:** Keep existing dark/light toggle; ensure integration with new components.  
  *Acceptance:* Theme persists across pages; toggle updates UI instantly.  
  *Files:* app/layout.tsx, components/theme/ThemeProvider.tsx.  
  *Effort:* S  
  *Priority:* P2

#### Story: Sidebar & Topbar

* [ ] **Implement Sidebar:** Create components/navigation/Sidebar.tsx that renders role‑specific menus; use collapsible groups for Campaigns, Content Studio, Calendar, Assets, Analytics and Settings.  
  *Acceptance:* Sidebar displays correct modules; toggling collapse works; active link highlighted.  
  *Files:* components/navigation/Sidebar.tsx, components/navigation/MenuItem.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Topbar with search:** Develop components/navigation/TopBar.tsx containing search bar, notifications icon and user avatar; clicking avatar shows dropdown with profile and logout.  
  *Acceptance:* Search invokes command palette; notifications icon displays count; clicking shows dropdown list; user avatar actions work.  
  *Files:* components/navigation/TopBar.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Command palette integration:** Adapt existing cmdk implementation to provide global search and AI commands; show results for campaigns, content and assets; support fuzzy search.  
  *Acceptance:* Typing command palette returns relevant results; selecting opens page or triggers AI generation; keyboard shortcuts work.  
  *Files:* components/navigation/CommandPalette.tsx, lib/hooks/useCommandPalette.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Notifications panel:** Build notifications dropdown using Radix UI; integrate with real‑time service; show content approvals, publication reminders and system announcements.  
  *Acceptance:* New events appear in panel; clicking notification navigates to relevant page; unread count resets when opened.  
  *Files:* components/navigation/Notifications.tsx, lib/hooks/useNotifications.ts.  
  *Effort:* M  
  *Priority:* P3

### Epic: Dashboard Modules

#### Story: Creator Dashboard

* [ ] **Summary cards:** Implement components/dashboard/SummaryCard.tsx to display metrics such as active campaigns, drafts awaiting approval, scheduled posts and total impressions.  
  *Acceptance:* Cards show correct counts; update when data changes; use icons and colours.  
  *Files:* components/dashboard/SummaryCard.tsx, app/(dashboard)/creator/page.tsx.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Campaign list:** Build components/dashboard/CampaignList.tsx for creators to view their campaigns; include status chips, due dates, and quick actions (edit, view, delete).  
  *Acceptance:* List fetches campaigns via API; sorting/filtering works; clicking opens details.  
  *Files:* components/dashboard/CampaignList.tsx, app/(dashboard)/creator/page.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Quick actions:** Add components/dashboard/QuickActions.tsx with buttons to create new campaign or content; open respective modals.  
  *Acceptance:* Buttons open forms; forms validated; creation persists via API; page refreshes list.  
  *Files:* components/dashboard/QuickActions.tsx, app/(dashboard)/creator/page.tsx.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Draft reminders:** Display list of content drafts requiring action; link to editor page; highlight overdue drafts.  
  *Acceptance:* Drafts appear sorted by last updated; clicking opens content editor; overdue drafts flagged.  
  *Files:* components/dashboard/DraftReminders.tsx, app/(dashboard)/creator/page.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **AI suggestions widget:** Provide small widget recommending content ideas based on trending topics; call AI suggestions endpoint.  
  *Acceptance:* Widget shows suggestions; clicking one populates Content creation form; loading state handled.  
  *Files:* components/dashboard/AISuggestions.tsx, app/(dashboard)/creator/page.tsx.  
  *Effort:* M  
  *Priority:* P2

#### Story: Brand Dashboard

* [ ] **Brand metrics cards:** Implement components/dashboard/BrandMetrics.tsx to show budgets spent, ROI, number of active campaigns and average engagement rates.  
  *Acceptance:* Data aggregated from DB; numbers update in near real time; cards clickable to view details.  
  *Files:* components/dashboard/BrandMetrics.tsx, app/(dashboard)/brand/page.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Approval queue:** Create components/dashboard/ApprovalQueue.tsx listing content awaiting approval; show thumbnails, summary and actions (approve/reject).  
  *Acceptance:* Items fetched from API; approving or rejecting updates status; list refreshes automatically.  
  *Files:* components/dashboard/ApprovalQueue.tsx, app/(dashboard)/brand/page.tsx, app/api/contents/[id]/approve/route.ts.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Budget vs ROI chart:** Use Recharts in components/dashboard/BudgetChart.tsx to visualise campaign spend against ROI over time.  
  *Acceptance:* Chart loads aggregated data; supports timeframe selection; responsive design.  
  *Files:* components/dashboard/BudgetChart.tsx, app/(dashboard)/brand/page.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Creator performance leaderboard:** Display a leaderboard ranking creators by engagement metrics (views, conversions).  
  *Acceptance:* Leaderboard updates in real time; sorting by metric works; clicking a creator opens their campaigns.  
  *Files:* components/dashboard/CreatorLeaderboard.tsx, app/(dashboard)/brand/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Campaign health summary:** Provide overview of active campaigns with status indicators (on track/behind schedule/over budget).  
  *Acceptance:* Summary shows each campaign with health indicator; clicking links to details; behind schedule flagged.  
  *Files:* components/dashboard/CampaignHealth.tsx, app/(dashboard)/brand/page.tsx.  
  *Effort:* M  
  *Priority:* P2

#### Story: Admin Dashboard

* [ ] **User management table:** Build components/admin/UserTable.tsx listing users (name, email, role, status, org); support search, pagination, sorting and toggling active status.  
  *Acceptance:* Admin can enable/disable users; changes persist; errors handled.  
  *Files:* components/admin/UserTable.tsx, app/(dashboard)/admin/page.tsx, app/api/users/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Organization management:** Create components/admin/OrgTable.tsx listing organisations; admin can rename or delete them; deletion requires confirmation and cascades data removal.  
  *Acceptance:* Deleting organisation removes associated campaigns, content and users (or disables them).  
  *Files:* components/admin/OrgTable.tsx, app/api/organizations/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Role management UI:** If using Role model, implement UI to create/edit roles and assign permissions; admin can assign roles to users.  
  *Acceptance:* Changing permissions updates DB; UI reflects changes; misuse prevented.  
  *Files:* components/admin/RoleManager.tsx, app/api/roles/route.ts.  
  *Effort:* M  
  *Priority:* P3
* [ ] **Audit logs:** Provide page to view audit logs (user actions, login attempts, content changes).  
  *Acceptance:* Admin can filter by user/date; logs stored in DB; sensitive data masked.  
  *Files:* components/admin/AuditLogs.tsx, app/api/logs/route.ts, schema.prisma (AuditLog model).  
  *Effort:* M  
  *Priority:* P3
* [ ] **Feature flags:** Implement toggle in admin settings to enable/disable experimental features (e.g. AI translation).  
  *Acceptance:* Flags stored in DB; toggles update UI accordingly; only admin can change.  
  *Files:* components/admin/FeatureFlags.tsx, app/api/features/route.ts, schema.prisma (FeatureFlag model).  
  *Effort:* M  
  *Priority:* P3

### Epic: Campaign Module

#### Story: Campaign List & Board

* [ ] **Campaign table:** Implement app/campaigns/page.tsx to display campaigns in table format; columns: name, status, start/end dates, budget, actions.  
  *Acceptance:* Users can sort by start date; filter by status; search by name.  
  *Files:* app/campaigns/page.tsx, components/campaigns/CampaignTable.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Pagination & infinite scroll:** Add pagination for campaigns; optionally support infinite scroll for board view.  
  *Acceptance:* Loading next page fetches more campaigns; user never loses context.  
  *Files:* components/campaigns/CampaignTable.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Board view:** Create app/campaigns/board/page.tsx using existing Kanban board component; columns represent statuses (Draft, Active, Completed); cards show campaign name, number of content items and due date.  
  *Acceptance:* Dragging a card updates campaign status in DB; dropping triggers optimistic update; board persists order.  
  *Files:* app/campaigns/board/page.tsx, components/campaigns/CampaignCard.tsx, components/board/Board.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Campaign detail page:** Implement app/campaigns/[id]/page.tsx with tabs: Overview, Content, Schedule, Analytics, Settings.  
  *Acceptance:* Page fetches campaign details via API; tabs load respective components; editing campaign updates DB.  
  *Files:* app/campaigns/[id]/page.tsx, components/campaigns/CampaignOverview.tsx, components/campaigns/CampaignTabs.tsx.  
  *Effort:* L  
  *Priority:* P0
* [ ] **Campaign creation wizard:** Develop multi‑step wizard at app/campaigns/new/page.tsx: Step 1 – basic info; Step 2 – budget & schedule; Step 3 – assign creators; Step 4 – review & create.  
  *Acceptance:* Progress indicator shows current step; validations at each step; hitting “Create” calls API; wizard resets on completion.  
  *Files:* app/campaigns/new/page.tsx, components/campaigns/CreateCampaignWizard.tsx, components/forms/CampaignForm.tsx.  
  *Effort:* L  
  *Priority:* P0
* [ ] **Campaign duplication UI:** Provide “Duplicate” button on campaign page; open modal to confirm duplication; call duplicate API.  
  *Acceptance:* New campaign appears in list; same content duplicated; schedules not copied.  
  *Files:* components/campaigns/DuplicateCampaign.tsx, app/campaigns/[id]/page.tsx.  
  *Effort:* S  
  *Priority:* P3
* [ ] **Campaign deletion confirmation:** Add confirmation dialog before deleting; require typing campaign name to confirm.  
  *Acceptance:* Campaign only deleted after confirmation; error messages shown if fails.  
  *Files:* components/campaigns/DeleteCampaignDialog.tsx, app/campaigns/[id]/page.tsx.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Campaign search:** Add search input above table; use debounced query to filter campaigns in API call; highlight search term in results.  
  *Acceptance:* Typing filters campaigns; network requests debounced; no flicker.  
  *Files:* components/campaigns/CampaignSearch.tsx, app/campaigns/page.tsx.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Campaign permissions UI:** Display read‑only state for users without edit permission; hide edit/delete buttons; show callout explaining permission.  
  *Acceptance:* Creator cannot edit brand‑owned campaigns; UI indicates limitations.  
  *Files:* components/campaigns/CampaignPermissions.tsx.  
  *Effort:* S  
  *Priority:* P2

### Epic: Content Module

#### Story: Content Editor

* [ ] **Rich text editor integration:** Add WYSIWYG editor (e.g. TipTap) to components/forms/ContentForm.tsx; support headings, lists, code blocks, images.  
  *Acceptance:* Users can format text; images inserted via asset picker; output saved in DB.  
  *Files:* components/forms/ContentForm.tsx, package.json.  
  *Effort:* L  
  *Priority:* P0
* [ ] **AI assistant panel:** Create components/ai/AssistantPanel.tsx that takes a prompt and calls AI generation endpoint; support suggestions, rewriting and tone adjustments.  
  *Acceptance:* Panel shows conversation; suggestions inserted into editor at cursor; loading states handled.  
  *Files:* components/ai/AssistantPanel.tsx, app/api/ai/generate/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Auto‑save drafts:** Implement auto‑save every 30 seconds or on blur; store draft state in DB with status draft; show “Last saved at …” indicator.  
  *Acceptance:* Draft persists across refresh; saving fails gracefully; unsaved changes prompt on navigation away.  
  *Files:* components/forms/ContentForm.tsx, app/api/contents/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Asset picker integration:** Add asset selector to content form; allow uploading new assets or selecting existing; show thumbnails; update assetId in content.  
  *Acceptance:* Selected asset preview appears; new uploads added to library; removal resets selection.  
  *Files:* components/forms/ContentForm.tsx, components/assets/AssetPicker.tsx, app/api/assets/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Content preview:** Provide side‑by‑side preview of formatted content; update live as user types.  
  *Acceptance:* Preview updates without lag; supports images and videos; accessible on small screens.  
  *Files:* components/forms/ContentPreview.tsx, app/content/[id]/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Translation & summarisation:** Add buttons in editor to translate content to selected language and summarise long posts using AI endpoints.  
  *Acceptance:* Clicking button inserts translated/summary text; credit usage tracked; UI shows cost.  
  *Files:* components/forms/ContentForm.tsx, app/api/ai/translate/route.ts, app/api/ai/summarise/route.ts.  
  *Effort:* M  
  *Priority:* P3
* [ ] **Status management:** Provide status dropdown (Draft, Submitted, Approved, Published, Rejected); restrict editing when approved or published; show status badge on content cards.  
  *Acceptance:* Changing status updates DB; UI reflects state; restrictions enforced.  
  *Files:* components/forms/ContentForm.tsx, components/content/ContentCard.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Comments and versioning:** Enable comments on content; store comment threads in DB; allow authors and reviewers to reply; implement version history to track edits.  
  *Acceptance:* Comments show user avatar and timestamp; editing content creates new version; users can view diffs between versions.  
  *Files:* components/content/CommentSection.tsx, app/api/comments/route.ts, schema.prisma (Comment and ContentVersion models).  
  *Effort:* L  
  *Priority:* P3

#### Story: Content List & Cards

* [ ] **Content list:** Develop components/content/ContentList.tsx to list content items within a campaign; includes status, type, author and actions (edit, delete, schedule).  
  *Acceptance:* List fetches content via API; sorting by status and createdAt; actions available per permission.  
  *Files:* components/content/ContentList.tsx, app/campaigns/[id]/page.tsx.  
  *Effort:* M  
  *Priority:* P0
* [ ] **Content card component:** Create components/content/ContentCard.tsx used in lists and search results; show excerpt, status badge and quick actions.  
  *Acceptance:* Card responsive; clicking card opens content editor; actions accessible via dropdown.  
  *Files:* components/content/ContentCard.tsx.  
  *Effort:* S  
  *Priority:* P0
* [ ] **Filtering & search:** Add filter controls (status, type, author) and search field to content list; update query parameters and API call accordingly.  
  *Acceptance:* Filtering reduces list; search matches title/body; update persists in URL for shareable views.  
  *Files:* components/content/ContentFilters.tsx, app/content/page.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Bulk actions:** Implement bulk select and actions (delete, submit, schedule) on content list.  
  *Acceptance:* Selecting multiple items shows action bar; operations performed on all selected items; success/failure notifications displayed.  
  *Files:* components/content/ContentList.tsx, components/content/BulkActions.tsx.  
  *Effort:* M  
  *Priority:* P2

### Epic: Calendar & Scheduling

#### Story: Calendar View

* [ ] **Calendar page:** Create app/calendar/page.tsx using react-day-picker or similar library; display scheduled posts as events; color‑code by platform.  
  *Acceptance:* Events appear on correct dates; clicking event opens schedule editing modal; can move events via drag‑and‑drop to reschedule.  
  *Files:* app/calendar/page.tsx, components/calendar/ScheduleEvent.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Schedule form modal:** Build components/calendar/ScheduleFormModal.tsx; used to create or edit a schedule; fields: content, platform, date, time, timezone.  
  *Acceptance:* Form validates input; saving calls API; modal closes on success; errors shown.  
  *Files:* components/calendar/ScheduleFormModal.tsx, app/calendar/page.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Timezone support:** Add timezone selection to user settings; convert scheduled times accordingly when displaying in calendar.  
  *Acceptance:* Times display in user’s timezone; scheduling respects timezone; stored in UTC.  
  *Files:* app/settings/page.tsx, lib/timezone.ts, components/calendar/ScheduleFormModal.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Recurring schedules:** Allow scheduling recurring posts (e.g. weekly recap); store recurrence rules; show series in calendar.  
  *Acceptance:* Recurring events created; editing one updates all or just that event; API handles recurrence rules.  
  *Files:* schema.prisma (RecurrenceRule model), app/api/schedules/recurring/route.ts, components/calendar/RecurringForm.tsx.  
  *Effort:* L  
  *Priority:* P3

### Epic: Assets Module

#### Story: Asset Library UI

* [ ] **Assets page:** Implement app/assets/page.tsx listing assets; include search, filter by type (image/video/doc); show thumbnails or icons.  
  *Acceptance:* Assets loaded via API; clicking asset shows details; uploading new asset adds to list.  
  *Files:* app/assets/page.tsx, components/assets/AssetList.tsx.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Asset upload dialog:** Create components/assets/UploadDialog.tsx triggered by “Upload Asset” button; allow multiple file selection; show progress bar for each file.  
  *Acceptance:* Upload progress shown; completed uploads appear in library; invalid files rejected.  
  *Files:* components/assets/UploadDialog.tsx, app/api/assets/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Asset detail drawer:** Implement side drawer to show asset details (preview, file name, size, uploaded by, usage count); provide option to delete if not in use.  
  *Acceptance:* Drawer opens on clicking asset; usage count fetched via API; deletion disabled if in use.  
  *Files:* components/assets/AssetDrawer.tsx, app/assets/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Asset usage indicator:** Show icons on content cards indicating attached assets; clicking opens preview.  
  *Acceptance:* Visual indicator appears; clicking opens asset drawer.  
  *Files:* components/content/ContentCard.tsx, components/assets/AssetPreview.tsx.  
  *Effort:* S  
  *Priority:* P2

### Epic: Analytics Module

#### Story: Analytics Dashboards

* [ ] **Overview chart:** Create components/analytics/OverviewChart.tsx to display overall engagement trends across all content (views, clicks, conversions) over time.  
  *Acceptance:* Chart loads with aggregated data; timeframe selector (7/30/90 days) works; tooltips show values.  
  *Files:* components/analytics/OverviewChart.tsx, app/(dashboard)/brand/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Campaign analytics tab:** On campaign detail page, build Analytics tab showing metrics specific to that campaign: line chart for views over time, bar chart for platform distribution, pie chart for content type breakdown.  
  *Acceptance:* Charts render with correct data; summary numbers displayed at top; loading state shown.  
  *Files:* components/analytics/CampaignAnalytics.tsx, app/campaigns/[id]/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Content analytics tab:** On content editor page, show metrics for that piece: total views, engagement rate, average reading time; support timeframe selection.  
  *Acceptance:* Data updates when timeframe changes; graphs and numbers reflect correct counts.  
  *Files:* components/analytics/ContentAnalytics.tsx, app/content/[id]/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Creator performance dashboard:** Build page for creators to view their performance metrics across campaigns: total views, average engagement, ROI.  
  *Acceptance:* Page summarises data; ranking relative to other creators; encourages improvement.  
  *Files:* app/analytics/creator/page.tsx, components/analytics/CreatorMetrics.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Export analytics UI:** Provide export buttons on analytics pages to download CSV/Excel; call export API; show loading and success states.  
  *Acceptance:* Clicking export triggers file download; file contains correct data; non‑admins cannot export.  
  *Files:* components/analytics/ExportButton.tsx, app/api/analytics/export/route.ts.  
  *Effort:* M  
  *Priority:* P3

### Epic: Settings Module

#### Story: User & Organization Settings

* [ ] **Profile settings:** Implement app/settings/profile/page.tsx; allow updating name, email, password; show API key usage (for AI).  
  *Acceptance:* Updates persist; email uniqueness enforced; password hashed.  
  *Files:* app/settings/profile/page.tsx, app/api/users/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Organization settings:** Create app/settings/organization/page.tsx; allow renaming organisation, updating slug, uploading logo, managing default language.  
  *Acceptance:* Changes persist to DB; slug unique; logo uploaded to assets library.  
  *Files:* app/settings/organization/page.tsx, app/api/organizations/[id]/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Notification preferences:** Provide UI to manage notification types (email, in‑app, push) per event (content approved, schedule published, comment).  
  *Acceptance:* Preferences saved in DB; notifications delivered accordingly.  
  *Files:* app/settings/notifications/page.tsx, app/api/notifications/preferences/route.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Billing & quotas:** Expose page for brand admins to view subscription plan, AI usage quotas and upgrade; integrate with payment provider (Stripe).  
  *Acceptance:* Page shows current plan and usage; upgrade flow works in test mode; proper error handling.  
  *Files:* app/settings/billing/page.tsx, app/api/billing/route.ts.  
  *Effort:* L  
  *Priority:* P3
* [ ] **API keys management:** Allow admins to generate and revoke API keys for integration; keys hashed in DB; show creation date and last usage.  
  *Acceptance:* Generating key returns token; revoking removes access; tokens stored hashed.  
  *Files:* app/settings/api-keys/page.tsx, app/api/api-keys/route.ts, schema.prisma (APIKey model).  
  *Effort:* M  
  *Priority:* P3

### Epic: Internationalisation

#### Story: Add i18n Support

* [ ] **Install next-intl:** Add next-intl dependency; configure i18n in next.config.ts with locales en and vi.  
  *Acceptance:* Build runs; locale detection works; fallback to en.  
  *Files:* package.json, next.config.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Create translation files:** Generate JSON files under locales/en.json and locales/vi.json containing translations for UI strings; start with common phrases.  
  *Acceptance:* UI components import and display translations; missing keys fallback gracefully.  
  *Files:* locales/en.json, locales/vi.json.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Update components to use translations:** Replace hardcoded strings in components with t('key') calls; update forms, buttons and messages.  
  *Acceptance:* All visible text translates when locale changes; no missing keys.  
  *Files:* all page and component files.  
  *Effort:* L  
  *Priority:* P2
* [ ] **Language switcher:** Add language selector in settings and optionally header; persisting choice in user profile.  
  *Acceptance:* Selecting language reloads page; preference saved to DB.  
  *Files:* components/navigation/LanguageSwitcher.tsx, app/settings/profile/page.tsx.  
  *Effort:* S  
  *Priority:* P2

### Epic: Testing & Quality

#### Story: Unit & Integration Tests

* [ ] **Set up Jest or Vitest:** Install jest, @testing-library/react, @testing-library/jest-dom, ts-jest or vitest. Configure jest.config.ts.  
  *Acceptance:* Running pnpm test executes tests; example test passes.  
  *Files:* package.json, jest.config.ts or vitest.config.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Write tests for utilities:** Cover lib/auth.ts, lib/rbac.ts, lib/utils.ts with unit tests.  
  *Acceptance:* Tests verify hashing, role checks and helper functions; achieve >80% coverage.  
  *Files:* tests/lib/auth.test.ts, tests/lib/rbac.test.ts, tests/lib/utils.test.ts.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Test forms & components:** Write RTL tests for LoginForm, RegisterForm, CampaignForm and ContentForm; validate form validation and submission calls.  
  *Acceptance:* Tests ensure form validation works; API calls triggered on submit; error states displayed.  
  *Files:* tests/components/forms/LoginForm.test.tsx, etc.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Test API routes:** Use supertest or Next.js test utilities to test API endpoints for auth, campaigns, contents, assets and schedules; mock Prisma.  
  *Acceptance:* Tests cover success and failure cases (403, 400, 404); ensure RBAC enforced.  
  *Files:* tests/api/campaigns.test.ts, etc.  
  *Effort:* L  
  *Priority:* P2

#### Story: End‑to‑End Tests

* [ ] **Setup Playwright:** Install @playwright/test and create e2e test suite; configure headless mode for CI.  
  *Acceptance:* Running pnpm e2e executes tests; base URL configured via environment variable.  
  *Files:* package.json, playwright.config.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Write e2e test for login:** Automate login flow; ensure user can log in and see dashboard; invalid login shows error.  
  *Acceptance:* Test passes on CI; failure indicates UI break.  
  *Files:* tests/e2e/login.spec.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Write e2e test for creating campaign:** Automate steps to create a campaign via wizard; verify creation success and board update.  
  *Acceptance:* Test passes; new campaign appears on board; details page accessible.  
  *Files:* tests/e2e/create-campaign.spec.ts.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Write e2e test for content workflow:** Simulate content creation with AI suggestion, submission and approval; verify status changes and notifications.  
  *Acceptance:* Test covers end‑to‑end flow; ensures integrations work; catch regressions.  
  *Files:* tests/e2e/content-flow.spec.ts.  
  *Effort:* L  
  *Priority:* P3
* [ ] **Write e2e test for schedule & publish:** Schedule content, run publish worker in test mode, verify content status updates to published.  
  *Acceptance:* Test passes; publish worker triggers; analytics event recorded.  
  *Files:* tests/e2e/schedule-publish.spec.ts.  
  *Effort:* L  
  *Priority:* P3

### Epic: CI/CD & Observability

#### Story: Continuous Integration

* [ ] **Lint & type check workflow:** Create .github/workflows/ci.yml that runs on PRs; install dependencies with pnpm; run pnpm lint, pnpm build, and pnpm test; fail builds on error.  
  *Acceptance:* Workflow executes on PR; fails if lint or tests fail; status visible in GitHub.  
  *Files:* .github/workflows/ci.yml.  
  *Effort:* S  
  *Priority:* P0
* [ ] **End‑to‑end workflow:** Add job to run Playwright tests using GitHub Actions service containers (Chromium) after build; store video/screenshots on failure.  
  *Acceptance:* E2E tests run in CI; failing tests upload artifacts; PR blocked.  
  *Files:* .github/workflows/e2e.yml.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Health check:** After build, run curl http://localhost:3000/api/health to ensure server starts and returns 200; fail workflow if not.  
  *Acceptance:* Workflow fails when health check fails.  
  *Files:* .github/workflows/ci.yml, app/api/health/route.ts (needs to be implemented).  
  *Effort:* S  
  *Priority:* P1

#### Story: Continuous Delivery & Deployment

* [ ] **Dockerfile:** Create Dockerfile to build and run Next.js app with Node 20; copy Prisma migrations; run prisma migrate deploy at startup.  
  *Acceptance:* Container builds successfully; runs server on port 3000; connects to DB via environment variable.  
  *Files:* Dockerfile.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Docker Compose:** Create docker-compose.yml for development containing app, postgres and (optional) mailhog; define volumes.  
  *Acceptance:* Running docker-compose up starts all services; app accessible; DB persistent.  
  *Files:* docker-compose.yml.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Deployment workflow:** Add .github/workflows/deploy.yml to build Docker image, push to registry and deploy to environment (e.g. AWS ECS or Vercel) on main branch push.  
  *Acceptance:* On merge to main, application deploys automatically; environment variables injected via secrets.  
  *Files:* .github/workflows/deploy.yml.  
  *Effort:* M  
  *Priority:* P3

#### Story: Observability & Monitoring

* [ ] **Logging:** Integrate pino or winston; create logger instance in lib/logger.ts; log requests, errors and important events in API routes.  
  *Acceptance:* Logs output to console in dev; structured JSON logs in production; sensitive data omitted.  
  *Files:* lib/logger.ts, all API routes.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Error tracking:** Add Sentry SDK; configure DSN via environment variable; capture exceptions in API and client.  
  *Acceptance:* Exceptions trigger Sentry reports; performance monitoring optional.  
  *Files:* lib/sentry.ts, app/layout.tsx, all API routes.  
  *Effort:* S  
  *Priority:* P3
* [ ] **Health endpoint:** Create /api/health route returning status, version and database connectivity; used in CI and monitoring.  
  *Acceptance:* GET returns {status: 'ok', version}; fails if DB unreachable.  
  *Files:* app/api/health/route.ts.  
  *Effort:* S  
  *Priority:* P1
* [ ] **Metrics collection:** Instrument API routes with Prometheus metrics using prom-client; expose /api/metrics endpoint; later integrate with Grafana.  
  *Acceptance:* Metrics endpoint exposes counters and histograms; accessible only in secure contexts.  
  *Files:* app/api/metrics/route.ts, lib/metrics.ts.  
  *Effort:* M  
  *Priority:* P3

### Epic: AI Integration

#### Story: AI Services

* [ ] **Select AI provider:** Evaluate available AI models (OpenAI GPT‑4, Cohere, local models) for content generation; decide based on cost, performance, languages.  
  *Acceptance:* Decision documented with pros/cons; API keys stored securely.  
  *Files:* docs/ai\_provider.md (create).  
  *Effort:* M  
  *Priority:* P1
* [ ] **Implement AI proxy endpoint:** app/api/ai/generate/route.ts calls selected AI provider; includes prompt engineering guidelines; handles errors and rate limiting; caches results.  
  *Acceptance:* Endpoint returns coherent content; rate limiting prevents abuse; caching reduces cost.  
  *Files:* app/api/ai/generate/route.ts.  
  *Effort:* M  
  *Priority:* P1
* [ ] **Prompt library:** Create file lib/prompts.ts containing reusable prompt templates for different content types (blog, tweet, caption, slogan); include placeholders for brand voice and tone.  
  *Acceptance:* Templates export functions that build prompts; used by AI assistant panel.  
  *Files:* lib/prompts.ts.  
  *Effort:* S  
  *Priority:* P2
* [ ] **AI usage tracking:** Add fields to AnalyticsEvent or create AIUsage model to record AI requests per user; display usage in settings; enforce quotas.  
  *Acceptance:* Each call to AI endpoint increments usage; quotas enforced; user sees remaining credits.  
  *Files:* schema.prisma, app/api/ai/usage/route.ts, app/settings/profile/page.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **AI tone tuning:** Provide UI controls to adjust tone (formal, friendly, professional) and style; include in prompt; update results live.  
  *Acceptance:* Changing tone alters generated content; UI intuitive; default tone stored in user settings.  
  *Files:* components/ai/AssistantPanel.tsx, lib/prompts.ts.  
  *Effort:* S  
  *Priority:* P3
* [ ] **Content idea generator:** Implement endpoint POST /ai/ideas to generate a list of campaign or content ideas given topics; integrate with dashboard widget.  
  *Acceptance:* Widget displays list of ideas; clicking inserts into campaign creation wizard; cost recorded.  
  *Files:* app/api/ai/ideas/route.ts, components/dashboard/AISuggestions.tsx.  
  *Effort:* M  
  *Priority:* P3

### Epic: Performance & Scalability

#### Story: Optimisations

* [ ] **Page size analysis:** Use next build --profile to analyse bundle size; identify large modules; implement dynamic imports.  
  *Acceptance:* Report shows reduced JS size; no large unused modules loaded.  
  *Files:* package.json scripts, dynamic imports in pages/components.  
  *Effort:* M  
  *Priority:* P3
* [ ] **Image optimisation:** Use Next.js Image component for all images; configure remote patterns for S3; implement lazy loading.  
  *Acceptance:* Images load responsively; no layout shifts; lighthouse score improves.  
  *Files:* all components displaying images.  
  *Effort:* S  
  *Priority:* P2
* [ ] **Pagination & virtualisation:** For lists with >100 rows, implement virtualised list (e.g. React Virtualised) to improve performance.  
  *Acceptance:* Scrolling remains smooth; memory usage low; tested with large datasets.  
  *Files:* components/campaigns/CampaignTable.tsx, components/content/ContentList.tsx.  
  *Effort:* M  
  *Priority:* P2
* [ ] **Caching & SWR:** Use React Query caching strategies; prefetch data on page load; use staleTime and cacheTime appropriately.  
  *Acceptance:* Data updates rarely cause refetch; offline support improved.  
  *Files:* all query hooks.  
  *Effort:* M  
  *Priority:* P2

## 10) CI/CD & Quality Gates

To maintain code quality and reliability, implement the following CI/CD and quality gates:

1. **Static Analysis:** Run ESLint and Prettier on every commit via Husky pre‑commit hook; enforce no unused variables, consistent import order and formatting.
2. **Type Checking:** Execute tsc --noEmit in CI to ensure type safety across the codebase. Fail builds on type errors.
3. **Unit & Integration Tests:** Use Jest/Vitest to run unit and integration tests; require at least 80% coverage for utilities and core logic.
4. **End‑to‑End Tests:** Run Playwright e2e tests on PRs and merges to main; fail pipeline on broken flows. Capture screenshots and videos for debugging.
5. **Build & Health Check:** Build the Next.js app in CI; after build, run a local instance and ping /api/health endpoint to ensure server starts correctly.
6. **Code Review:** Enforce branch protection rules requiring at least one approval and passing status checks before merge. Use Conventional Commits for commit messages.
7. **Secrets Management:** Store sensitive values (DB connection strings, API keys) as GitHub repository or organisation secrets; never commit secrets to the repo. Provide .env.example template.
8. **Deployment Pipeline:** After tests pass on main, automatically build Docker image and deploy to staging environment; run smoke tests; promote to production on manual approval.

## 11) .env & Secrets Handling

1. **.env.example:** Provide a .env.example file listing all required environment variables with descriptions. Example variables:

* DATABASE\_URL="postgresql://aim\_user:password@localhost:5432/aim\_db"  
  NEXTAUTH\_SECRET="set-a-random-secret-here"  
  NEXTAUTH\_URL="http://localhost:3000"  
  S3\_BUCKET="aim-assets"  
  S3\_REGION="ap-southeast-1"  
  S3\_ACCESS\_KEY\_ID=""  
  S3\_SECRET\_ACCESS\_KEY=""  
  AI\_API\_KEY=""  
  PUBLIC\_SITE\_URL="http://localhost:3000"  
  DEFAULT\_LANGUAGE="en"  
    
  # Optional  
  SMTP\_HOST=""  
  SMTP\_PORT=""  
  SMTP\_USER=""  
  SMTP\_PASS=""

1. **Local secrets:** Each developer should copy .env.example to .env and fill in values for their environment. .env must be included in .gitignore.
2. **Production secrets:** Use environment variables managed by deployment platform (Vercel, AWS, Docker secrets). Never commit secrets to git or store them in plain text. Rotate keys periodically and restrict permissions according to least privilege.
3. **Secret injection:** Use GitHub Actions secrets in workflows; avoid logging secret values. For NextAuth, set NEXTAUTH\_URL and NEXTAUTH\_SECRET accordingly for each environment.

## 12) Open Questions

1. **Stack Upgrade Tolerance:** Should we strictly keep to Next.js 15 and current dependencies, or are we allowed to upgrade/downgrade versions (e.g. Next.js 14 LTS) to improve stability? This affects long‑term maintenance and compatibility.
2. **Additional Roles:** Beyond Creator, Brand and Admin, do we need Reviewer and Publisher roles? Understanding additional responsibilities helps refine RBAC and UI flows.
3. **Compliance Requirements:** Are there specific regulatory or compliance requirements (e.g. GDPR, ISO 27001) that we need to satisfy? This influences data retention, consent management, logging and audit features.

## 13) Appendix

### Proposed Scripts & Commands

* pnpm prisma generate – Generate Prisma client after modifying schema.
* pnpm prisma migrate dev --name <migration> – Create and apply migrations in dev.
* pnpm seed – Run the seed script to populate initial data.
* pnpm dev – Start development server with local DB; uses Next.js with HMR.
* pnpm build – Build production assets; run type check and lint before build.
* pnpm test – Run unit and integration tests.
* pnpm e2e – Run Playwright end‑to‑end tests.
* node scripts/publish-worker.js – Run the scheduled publish worker (cron job).

### Folder Conventions

* app/ – Next.js app directory with app router; split into role‑based groups (auth), (dashboard), campaigns, content, calendar, assets, settings, analytics.
* components/ – Reusable UI components; organised by domain (navigation, dashboard, campaigns, content, calendar, assets, analytics, forms, ai).
* lib/ – Shared utilities (Prisma client, RBAC helpers, logger, metrics, prompts, hooks).
* prisma/ – Database schema and migrations.
* public/ – Static assets and icons.
* scripts/ – Utility scripts (seed, cron jobs, test data generation).
* tests/ – Unit, integration and e2e tests; mirror project structure.

### Coding Guidelines

* Use **TypeScript** everywhere; avoid any. Prefer explicit types and generics.
* Follow **Conventional Commits** for commit messages (feat:, fix:, chore:, refactor: etc.). This enables automated changelogs.
* Use **ESLint** with Next.js preset and Prettier. Keep code clean and consistent.
* Prefer **server components** for pages and components that do not require client interactivity; use client components only when needed (forms, interactive UI, stateful logic).
* Use **React Query** for data fetching and caching; colocate queries with components but abstract into hooks for reuse.
* Validate all inputs with **zod** at the boundary (API routes, forms) and handle errors gracefully.
* Keep **UI accessible** by using Radix UI primitives correctly and adding ARIA labels; ensure keyboard navigation works across components.
* Write **tests** for new components and API routes; aim for high coverage on business logic; prioritise critical paths.

### Sample .env.example

See Section 11 for the detailed .env.example content; copy and adjust for your environment. Ensure this file is kept up to date as new environment variables are introduced.

[[1]](https://github.com/rymleon23/ioqwfoihas/blob/master/README.md#L11-L18) GitHub

<https://github.com/rymleon23/ioqwfoihas/blob/master/README.md>

[[2]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L34-L57) [[3]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L1-L74) [[4]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L40-L42) [[5]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L14-L34) [[6]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L51-L54) [[7]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L55-L57) [[8]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L47-L57) [[9]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L50-L51) [[10]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L34-L56) [[11]](https://github.com/rymleon23/ioqwfoihas/blob/master/package.json#L60-L71) GitHub

<https://github.com/rymleon23/ioqwfoihas/blob/master/package.json>