ChatGPT

Himalytix ERP Project Status & Roadmap

Current Project Status

Overview – Himalytix ERP is a multi-tenant platform built with Django, PostgreSQL and Redis. It targets small and mid-sized enterprises, providing accounting, inventory, and user-management modules. The codebase implements a modern voucher entry experience and tenant-aware APIs, with opinionated defaults tuned for Nepalese deployments [1]. The repository is structured around a core `ERP/` Django project, documentation under `Docs/`, and legacy migration bundles [2].

Implemented Modules

- **Accounting** – Comprehensive journal entry system with configurable voucher templates, posting workflow, chart of accounts and general ledger. Models cover fiscal years, accounting periods, chart of accounts, journals, journal lines, dimensions (cost centre, department, project), tax and currency management [3]. Business logic is encapsulated in services for creating and posting vouchers, generating trial balance, closing periods and auto-numbering [4]. Views leverage HTMX for interactive voucher entry; form schemas are defined in YAML to enable dynamic UI generation [5].

- **Inventory** – Product catalog, warehouses, batch/serial management and stock snapshots tied back to GL accounts [6]. The module integrates with accounting to reflect cost of goods sold and inventory valuations.

- **Multi-Tenancy & Security** – Active tenant middleware sets the `search_path` for PostgreSQL schemas per request, ensuring strict tenant isolation [7]. Tenant-aware APIs and scoped permissions are provided via DRF; the system uses JWT and session authentication with rate limiting and role-based access control [8].

- **User Management** – Organization and user administration, role assignment, and tenant branding (per-tenant favicons) [9]. Supports multi-factor authentication, JWT, OAuth and row-level permission checks [10].

- **API Layer** – Versioned REST API under `/api/v1/` with OpenAPI schemas and Swagger/Redoc documentation [11]. The API architecture uses DRF viewsets with layered middleware for rate limiting, authentication, permissions and versioning [12].

- **Observability & DevOps** – Metrics via Prometheus and Grafana, structured logging, and distributed tracing with OpenTelemetry and Jaeger [13]. CI/CD pipeline uses GitHub Actions to lint, test, build Docker images, deploy to staging, run smoke tests, and deploy to production after manual approval [14].

- **Vertical for LPG/NOC Distribution** – Industry-specific layer with cylinder types, dealer management, logistics, and NOC purchase allocation; APIs under `/api/lpg/` [15].

Work in Progress / Pending

- **Phase 2 Accounting Enhancements** – Additional voucher types (e.g., recurring entries), year-end closing routines and auto-revaluation. Tests are partially in place (`test_phase2_views` and `test_models`) [16].
- **HR & Payroll** – The HR module is not fully implemented; basic employee and attendance tracking is anticipated but absent.
- **CRM & Project Management** – There is no dedicated CRM or project management module yet; these must be built to align with SAP-like scope.
- **Extensibility & Partner Hooks** – Event hooks and extension framework are limited; plugin architecture needs refinement to allow external apps to add fields or logic without modifying core code.

Overall Project Structure

Architecture Layers

- **Client Layer** – Web browser UI built with HTMX/Alpine.js and a potential mobile app using the REST API [17].
- **Load Balancer** – Nginx/HAProxy handles incoming traffic [18].
- **Application Layer** – Django web server (Gunicorn) processes requests, while Celery workers and schedulers handle background tasks [19].
- **Data Layer** – Primary and replica PostgreSQL databases, with Redis for caching and broker tasks [20].
- **Observability & External Services** – Prometheus, Grafana and Jaeger for metrics and tracing; S3/GCS for backups; SMTP for email; OAuth providers for social login [21].

Multi-Tenancy Model

Himalytix uses a **schema-per-tenant** approach: a public schema contains `tenants` and `users` tables; each tenant has its own schema containing business tables like `journal_entries`, `accounts` and `transactions` [7]. The middleware extracts the tenant from the subdomain or header, sets `search_path` accordingly and routes queries to the correct schema [22]. This ensures physical data isolation, simplifies backup/restore per tenant and enforces row-level security.

Accounting Module Architecture

The accounting module follows a clean separation of concerns:

- **Models** define fiscal periods, chart of accounts, journals and lines, dimensions, taxes and currencies [3]. `VoucherModeConfig` and YAML schemas control dynamic form generation [23] [5].
- **Views** are layered with mixins enforcing user organization, permissions and voucher configuration; HTMX is used for responsive updates [24].

- **Services & Repositories** encapsulate business logic such as creating and posting vouchers, trial balance generation, and closing periods [25]. The repository pattern decouples data access from business logic to enhance testability [26].
- **API** endpoints expose accounting operations via DRF viewsets and custom API views like `BulkJournalActionView` [27].

Repository Layout

- `ERP/` – Django project with apps for accounting, inventory, user management and API.
- `ERP/accounting` – Contains `models.py`, `views/`, `services/`, `repositories/`, `forms/`, `schemas/`, `templates/` and tests [28].
- `ERP/usermanagement` – Handles authentication, authorization and tenant branding.
- `ERP/Inventory` – Manages stock, warehouses and product catalog.
- `Docs/` – Architecture documentation and runbooks (deployment rollback, scaling, incident response) [29].
- `ERP/docs/adr/` – Architectural decision records such as multi-tenancy patterns.

Estimated Cost to Date

Himalytix ERP has been under active development since early 2024. Building a multi-tenant ERP with accounting, inventory and voucher entry requires substantial effort:

- **Development Team** – Assume a core team of 3–4 developers, one architect and one QA engineer. Roughly 8 months of effort have been invested to reach the current state (February–October 2024) at ~160 hours per person per month.
- **Total Effort** – 3.5 full-time equivalents × 8 months × 160 hours ≈ **4,480 hours** of work.
- **Cost Estimation** – At an average loaded cost of USD $50 per engineering hour (blended rate for developers, QA and architect), the project has consumed approximately **$224,000**. This estimate excludes infrastructure costs, which are minimal during development due to local Docker usage and cloud credits.

These values are indicative; actual costs vary based on team composition and local salary rates.

Comparison to SAP Business ByDesign

SAP Business ByDesign offers comprehensive finance, CRM, HR, supply chain, procurement and project management modules [30] [31]. It emphasises configuration-driven behaviour, document/voucher frameworks, posting logic and strong audit trails. Himalytix aligns with several of these patterns but currently focuses on accounting and inventory.

Area	SAP Business ByDesign [30]	Himalytix ERP
Finance	Full suite: GL, AP/AR, cash management, compliance & reporting [30]	GL with journal entries and posting workflow; AP/AR and cash management planned for Phase 2.

Area	SAP Business ByDesign [30]	Himalytix ERP
CRM	Marketing, sales and service modules [32]	Not yet implemented.
Human Resources	Time & labor management, payroll, workforce administration [33]	Planned; currently only user management.
Supply Chain Management	Product development, planning, manufacturing, warehousing and logistics [31]	Basic inventory management; manufacturing/production not implemented.
Procurement	Sourcing and purchasing [34]	Purchasing flows integrated in accounting (voucher types) but not full supplier management.
Project Management	Integrated project planning and execution [35]	Not available.
Industry-specific	Extensions for professional services, manufacturing, wholesale distribution [36]	LPG/NOC vertical implemented with custom workflows.
Configuration & Extensibility	Highly configurable document types, posting rules, voucher templates, and user interfaces	YAML-driven voucher schemas, dynamic form generation and model-based configuration provide similar flexibility.
Audit & Compliance	Strong audit trails, separation of entry and posting, status-driven workflows	Posting engine ensures draft/posted status, audit logs and reversal flows; further audit features in development.
Multi-Tenancy	Typically single-tenant SaaS; not schema-per-tenant	Schema-per-tenant model ensures strong data isolation.

Himalytix is on track to replicate SAP-like patterns in finance and to extend into other modules. However, several modules (CRM, HR, full procurement and project management) remain to be built to achieve parity.

MVP Deliverables & Refinement

The **Minimum Viable Product (MVP)** should aim to deliver a cohesive, auditable platform targeting SMEs with basic accounting and inventory needs. To minimise scope creep and align with enterprise patterns, the MVP should include:

1. **Core Platform**
2. Schema-per-tenant multi-tenancy with tenant provisioning, selection and management.
3. Role-based and object-level permissions, JWT/session authentication and tenant branding.
4. REST API v1 with OpenAPI documentation, Swagger UI and rate limiting.

5. **Accounting Module**

6. Chart of accounts with depth and sibling constraints and hierarchy editing.
7. Journal entry and posting engine with balanced debit/credit checks, fiscal year and period management, and reversal functionality.
8. Voucher templates defined via YAML schemas; dynamic UI generation with HTMX; support for attachments and approval workflows.

9. Basic AP/AR capability (vendor/customer master, invoice voucher type) and cash/bank ledger integration.

10. **Inventory Module**

11. Product catalog and categories, units of measure and SKU variants.
12. Warehouse management with stock in/out, batch/serial tracking, and inventory snapshots.

13. Inventory posting to GL accounts; integration with purchase vouchers and sales vouchers.

14. **User & Org Management**

15. User creation, role assignment, multi-factor authentication and password policies.

16. Organization creation, schema provisioning and per-tenant configuration.

17. **Industry Vertical (Optional)**

18. LPG/NOC vertical to serve an immediate target market; includes cylinder management, dealer credit enforcement and trip management [15].

19. **Observability & DevOps**

20. CI/CD pipeline with automated tests and linting; Docker Compose for local development.
21. Metrics, logging and tracing as detailed in the architecture documentation [13].

Refinements Needed:

- Expand test coverage across modules and edge cases, particularly for fiscal year closing and multi-tenant query routing.
- Implement AP/AR ledger and basic cash management to cover more finance scenarios.
- Build user-defined dimensions and reporting screens (trial balance, ledger, inventory age).
- Finalize plugin architecture (event hooks) for future CRM/HR modules.

Go-to-Market (GTM) Strategy

1. **Target Market & Positioning**
2. Focus on SMEs in Nepal and similar emerging markets needing affordable, compliant accounting software. Emphasize multi-tenant deployment for accounting firms serving multiple clients.

3. Highlight the LPG/NOC vertical as a unique differentiator for distributors and gas companies.

4. **Pricing Model**

5. Subscription per organization with tiered pricing based on user count and modules enabled.

6. Offer free tier or extended trial for up to 2 users and limited voucher entries to drive adoption.

7. **Sales & Partnerships**

8. Partner with local accounting firms and business consultants to distribute the platform; provide training and revenue share.

9. Engage with Nepalese regulators and tax authorities to ensure compliance and garner endorsement.

10. **Marketing Channels**

11. Content marketing: publish tutorials on journal entry workflows and inventory management; leverage community platforms and webinars.
12. Social media targeting SMEs; highlight success stories from early adopters in the LPG sector.

13. Attend local trade shows and business expos to demonstrate the platform.

14. **Customer Onboarding & Support**

15. Provide guided onboarding with pre-configured chart of accounts and voucher templates for common industries.

16. Offer in-app help, knowledge base and support channels. Leverage the schema-per-tenant model to sandbox demos.

17. **Roadmap to Full SAP-Like Suite**

18. Post-MVP, prioritise development of CRM, HR and procurement modules to broaden product appeal and close gaps vs SAP Business ByDesign.
19. Introduce project management and manufacturing modules in subsequent releases.
20. Continue refining configuration and extensibility to allow industry-specific verticals without code modifications.

Conclusion

Himalytix ERP has established a solid, auditable foundation with multi-tenant accounting and inventory management. The architecture adheres to enterprise patterns such as configuration-driven voucher entry, posting workflows and schema-per-tenant isolation. To compete with comprehensive solutions like SAP Business ByDesign, the project must expand into CRM, HR, procurement and project management while maintaining backward compatibility and configurability. A focused MVP, coupled with a strategic

go-to-market approach targeting SMEs and niche verticals, will create momentum and attract early adopters.

---

1 2 6 9 11 15 16 README.md
https://github.com/ritsnep/HimalytixNew/blob/HEAD/README.md

3 4 5 23 24 25 26 27 28 accounting_architecture.md
https://github.com/ritsnep/HimalytixNew/blob/HEAD/ERP/accounting_architecture.md

7 8 10 12 13 14 17 18 19 20 21 22 29 ARCHITECTURE.md
https://github.com/ritsnep/HimalytixNew/blob/HEAD/ERP/docs/ARCHITECTURE.md

30 31 32 33 34 35 36 SAP Business ByDesign | Features
https://www.sap.com/products/erp/business-bydesign/features.html