ENTERPRISE APPLICATION PLATFORM · LICENSED

The platform underneath your next hundred applications.

Business applications are declared, not engineered. Zorbit is a domain-agnostic, multi-tenant, configuration-first platform that you license and compose, module by module.



PLATFORM	LICENSING	CONTACT
Zorbit, by Utunga	Ask.	hello@utunga.com

THE IDEA

Implemented once. Inherited everywhere.

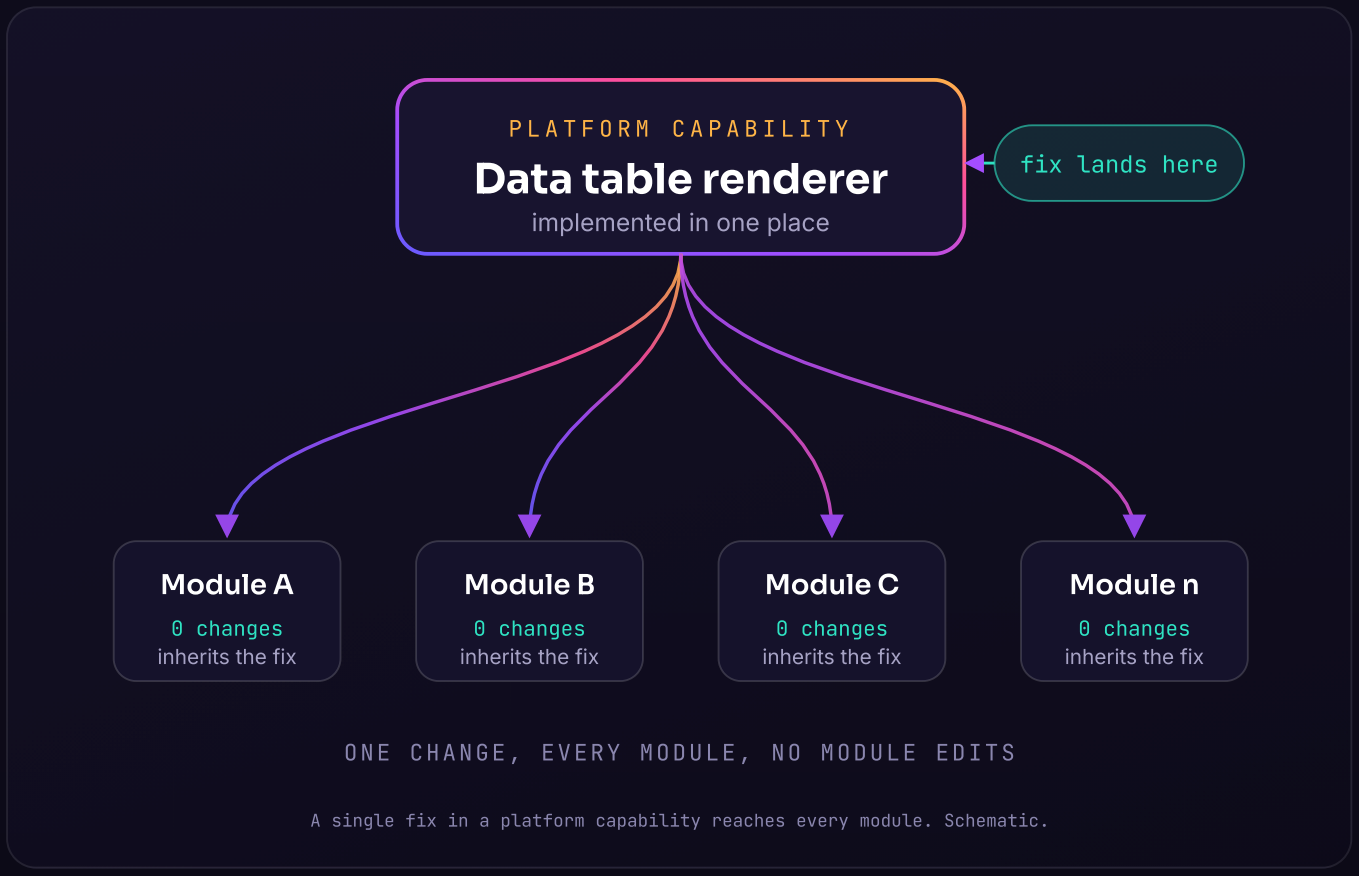
Every capability in Zorbit exists in exactly one place. Tables, forms, CRUD, workflow, documents, identity, secrets, personal-data protection and navigation are built once and refined centrally.

When a fix lands in the platform, every module has it. No module is edited.

A module that reimplements a platform capability is a defect, even if it works. That is the supreme principle of the design.

Business applications are declared, not engineered. A module states its intent in a manifest. Platform modules execute that intent.

The result is a platform you license once and compose many times.



A single fix in a platform capability reaches every module. Schematic.

> Business modules declare intent. Platform modules execute intent.

1	**20**	**100–250**	**0**
place every capability is implemented	non-negotiable design principles	modules, the design scale	lines of UI code per module

WHAT EVERY MODULE GETS

A catalogue on day one, not a feature dump.

Every service is consumed by declaration. Modules never implement these, so there is nothing to build, test or maintain per module.

RENDERERS

Data tables

List and card views, server-side paging, type-aware filters, saved filters, bulk actions, export.

Forms

Multi-step wizards, conditional questions, validation, a visual designer, versioned definitions.

CRUD and ingestion

Full create, read, update and delete from configuration. One ingestion sequence for forms, APIs and bulk files.

Detail and actions

Detail views and action buttons, addressed by capability notation. No screen code.

Guides

Per-module guides rendered by the platform from declared content.

SERVICES

Documents and PDF

Styled documents from declared data, with masked-data enforcement.

E-signing

Digital signing and verification workflows.

Notifications

Email, SMS, push and in-app, declared per module.

KYC

Provider catalogue and evidence capture behind one provider contract.

Analytics

One service, many datasets, configured per dataset.

AUTOMATION

Workflow

BPMN 2.0 processes plus queue and pipeline worklists, with SLA timers and escalation.

Rules engine

Deterministic rules on the workflow filter grammar. Configured, not coded.

AI gateway

One route for every LLM call: quotas, provider routing, full prompt audit.

Voice

Speech-to-text and text-to-speech, plus a record, extract, confirm, ingest pipeline.

Journey tester

Portable end-to-end journey bundles that run against the live console.

EIGHT SHIPPED RENDERERS

`DataTableView` `FormRenderer` `DetailView` `WorkflowQueueView` `GuideRenderer` `ActionButtonRenderer` `WizardRenderer` `EmptyStateRenderer`

Every screen is a platform renderer addressed by capability notation. About 90% of screens are a data table or a form.

ALSO IN THE PLATFORM

`File storage and viewer` `Document digitisation` `Data ingestion` `Realtime and chat` `Fraud detection engine` `Payment gateway` `Event bus` `Audit trail` `Observability` `Database console`

ARCHITECTURE

Five layers. Configuration over code.

Dependencies point one way, down. Business modules sit on top and state what they are.

Business modules	Declare intent. Author no UI and no CRUD code.
Extension services	Tables, forms, workflow, documents, AI gateway, voice, KYC and more.
Core	Identity, authorisation, registry, console, vaults, entitlement.
SDK	The module-facing toolkit.
Contracts	Manifest, envelope and identity contracts.

What a module is made of

- **Manifest.** Routing, resources, privileges, navigation, guide content and exports. Invalid manifests fail loudly.
- **Declared schemas.** Fields, persistence target and privileges per resource.
- **Declared workflows.** Filters, queues, pipelines and stage modes.
- **Declarative seeds.** Forms, tables and workflows ship as configuration.
- **Zero custom UI.** Every screen is a platform renderer.



90% configuration, 10% code

A design target for each business module, not a measurement.

The module factory

01	02	03	04	05
Describe	**Generate**	**Stage**	**Approve**	**Register**
Resources, fields, renderers, navigation, privileges, workflows.	A working module, no per-module UI or CRUD code.	Artefacts wait in staging.	Promoted only on human approval.	Atomic swap by version supersession.

CAPABILITY NOTATION

`@platform:DataTableView` `@platform:FormRenderer` `moduleId:Component`

One notation for every declared capability, offered automatically in the module builder. Code, configuration and credentials are three separable planes.

REGISTRY AND CONSOLE

Register a manifest. It appears in the console.

A manifest-only module shows up with zero code changes. Deregister it and it is gone.

01
Register
Manifest validated. Signed self-announcements accepted.

02
Ingest privileges
Declared privileges become platform privileges.

03
Extract navigation
Ingested once as cacheable fragments. Never read at serve time.

04
Lazy-load
Each module is a bundle, fetched on first menu click.

ONE MENU PER PERSON

`Registry navigation` × `User's privileges` × `Tenant entitlement` = `Per-user nav.json`

Cached in the browser and invalidated by content hash. Menu families, sections and editions are database rows.

What the console gives you

One shell, every module
A lazy-loaded shell. Renderers load once, centrally. The registry serves capability discovery in @platform: notation.

Impersonation with audit
A user switcher of permitted identities, a persistent indicator, an acting-as panel and one-click return. Both identities are audited.

Live refresh
Updates arrive over a WebSocket channel. No polling.

Closed route lists
Each menu leaf carries the closed list of backend routes its screen may call.

SHELL FEATURES

`Tree and minimal line styles` `Live search` `Breadcrumb path` `Keyboard shortcuts`
`Dark and light themes` `Resizable sidebar` `Edition selector` `Role badge` `Version footer`
`Per-user saved state` `Up to 10 levels`

MODULE LIFECYCLE

`● PENDING` — `● READY` — `● RETIRED`

A module becomes ready once its declared dependencies are ready.

IDENTITY AND DATA PROTECTION

Four layers, built in once.

Raw personal data lives in a physically separate database. Raw secrets never appear in manifests, logs or screens.

LAYER 1

Identity

A proven open-source IdP broker behind a provider contract. Realms map to tenants.

`OIDC` `SAML` `Social login` `LDAP` `Passkeys` `TOTP` `Magic link` `Step-up`

LAYER 2

Authorisation

Owned by Zorbit, never outsourced. Privileges are declared by modules; roles bundle them.

`Privilege catalogue` `15 system roles` `Org cascade` `Entitlement` `Namespace guard` `IP allowlist`

LAYER 3

Data protection

Personal data and secrets live in separate vaults. Everything else carries tokens.

`PII vault` `Tokenise every path` `Default-deny detokenise` `Secret vault` `Envelope encryption` `Hash-chain audit`

LAYER 4

Tenant data

Isolation in two independent layers, so one mistake does not cross a tenant boundary.

`Multi-tenant Postgres` `Tenant-scoped access layer` `Row-level security` `Audited system scope`

One ingestion sequence for every path

01 **Validate** — Declared schema.

02 **PII label check** — Likely PII is flagged.

03 **Tokenise** — Raw values go to the vault.

04 **Check** — Namespace and privilege.

05 **Write** — Operational store, declared events.

Forms, APIs, bulk Excel and CSV, integrations. Toggled per module in the manifest.

OPERATIONAL

Tokens and masks

Tokens, nicknames and masks. Never the raw value.

VAULT

PII database

Physically separate. Detokenisation is default-deny, per privilege and purpose, and audited.

OBJECTS

Object store

Masked content only.

WORKFLOW AND DOCUMENTS

Three layers of workflow, all declared.

A process engine, human worklists and system automation are separate layers. Each one is configured, not coded.

LAYER 1

BPMN 2.0 process engine

Long-running business processes, declared.

LAYER 2

Queue and pipeline worklists

Filters, queues and pipelines with maker-checker-approver chains.

LAYER 3

Integration automation

System-to-system automation and outbound connectors with secret references.

STAGE MODES

HUMAN · MODERATED · TOP-MODERATED · AUTOMATED

Maker, checker and approver chains with SLA timers and escalation. A deterministic rules engine uses the same filter grammar.

Documents, voice and AI

Documents and PDF

Styled documents from declared data. Masked-data enforcement means raw personal data never renders.

Digitisation and signing

OCR and structured extraction into declared resources. Digital signing and verification workflows.

AI gateway

One route for every LLM call: provider routing, secret-reference keys, quotas, full prompt and response audit. Raw personal data never enters a prompt.

Notifications and analytics

Email, SMS, push and in-app, declared per module. One analytics service, many datasets.

The voice pipeline

01 **Record** — Speech in.

02 **Transcribe** — Speech to text.

03 **Extract** — Structured fields.

04 **Stage** — Held for review.

05 **Confirm** — A person decides.

06 **Ingest** — Normal ingestion path.

A person confirms before anything is ingested. The result enters the normal ingestion path.

EXTENSIBILITY AND WHITE LABEL

Any new module. Your brand on all of it.

New modules need no platform changes. New tenants need no rebuild.

Extensibility

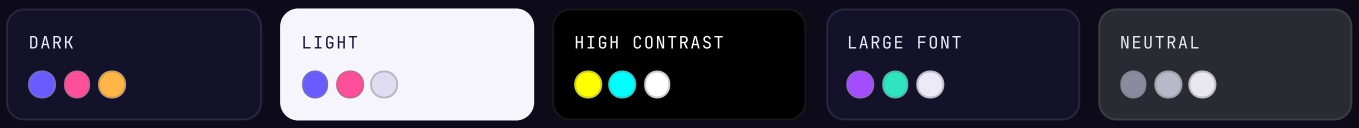
- A manifest-only module appears in the console with zero code changes.
- Deregister a module and it is removed.
- A module can be regenerated and swapped atomically, valid when the full test suite passes.
- Entitlement, what a tenant bought, is a separate core module. Two tenants with one role see different menus.
- Extend with new modules, not platform forks.

White label

- Per-tenant logo, logo text and title, plus a whole-application restyle on activation.
- Multiple themes per organisation. One is active.
- Effective theme resolves from organisation brand, per-user override and platform default.
- Themes resolve by domain or subdomain.
- Applied at runtime through CSS variables. No rebuild.

A theme is a closed set of 24 design tokens

`primary` `accent` `background` `surface` `card` `text 1` `text 2` `text 3` `border` `divider`
`sidebar x5` `active` `hover` `focus` `shadow x2` `radius x2` `density` `font`



Preloaded themes, with default, festival and occasion records. Swatches are illustrative.

How the effective theme resolves

01 **Domain or subdomain**
Finds the tenant.

02 **Organisation brand**
The active theme.

03 **Per-user override**
If the user set one.

04 **Platform default**
The fallback.

AI-assisted theme designer

A theme designer with AI-assisted generation extends the same token set. Nothing outside the 24 tokens can drift from the platform.

PROOF AND QUALITY

How we know it works.

Proven end to end on a regulated financial-services edition.

01

Demoable or not done

Every capability defines a demo path, a URL and click path followable in under a minute, with archived browser evidence.

02

Test-first journey bundles

Acceptance journeys are written before code, in self-installing, cross-platform JSON bundles with credentials externalised. Zero tests is banned.

03

13 test dimensions

Each capability is measured across thirteen dimensions, not just a happy path.

04

An integration gate

Nothing reaches production without passing the gate, and the gate is recorded.

The promotion path

DEV · SIT · QA · UAT · STAGE · PROD · DR

Seven stages from dev to disaster recovery. A blast-radius run over every dependant of a changed module records breaches in a ledger. Promotion to prod and DR stops at a human gate.

Independent scoring

01
Decompose
An architect splits work into dependency-ordered tasks.

02
Score
A deterministic rubric scores each task.

03
Plan tests
An independent planner writes the scenarios first.

04
Build
A builder matched to the score implements.

05
Verdict
An independent runner, never the builder, decides on evidence.

Hard failure, not silent failure

Missing adapters, invalid manifests, unsafe paths, undeclared capabilities and out-of-allowlist secret or PII access fail loudly. No shim, alias or hidden bridge survives testing without a bounded, expiring approval.

WHERE ZORBIT SITS

Build it, buy a category, or license the platform.

Most of this list is invisible until a buyer asks for it. Each row is something your team would own for as long as the platform lives.

CAPABILITY	BUILD IN-HOUSE	LOW-CODE PLATFORMS	VERTICAL SUITES	ZORBIT
Multi-tenant by design	yours to build	partial	partial	yes
Configuration-first modules	yours to build	partial	partial	yes
One implementation per capability	yours to enforce	partial	varies	yes
Module registry and single console	yours to build	partial	varies	yes
PII and secret vaults built in	yours to build	varies	varies	yes
End-to-end tester bundled	yours to build	separate tool	separate tool	yes
Vertical neutral	yes	yes	no	yes
Extend with new modules, not platform forks	yours to build	partial	vendor-led	yes
Time to first module	after the platform	fast to start	vendor roadmap	a manifest

Category examples only. Low-code platforms such as OutSystems, Mendix, Appian. Vertical suites such as the large policy or core-banking suites. Values describe categories, not named vendors.

BUILD IN-HOUSE

Total control, total ownership

Every row above is yours to build and then maintain.

LOW-CODE PLATFORMS

Fast to start

Quick to a first screen. Coverage of the wider platform concerns varies.

VERTICAL SUITES

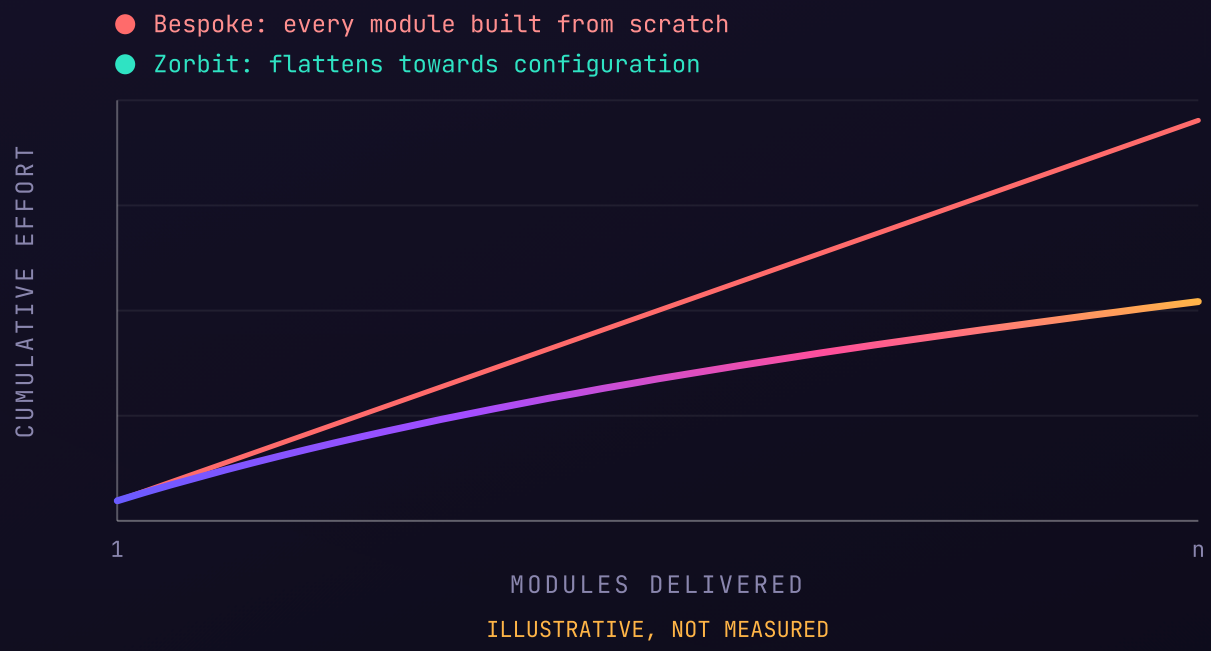
Fast if you are the vertical

Tied to one domain. Extension is led by the vendor.

ECONOMICS

The cost of the second module is the point.

Built bespoke, each module carries its own tables, forms, workflow and security. On Zorbit, those are inherited, so each new module trends towards configuration only.



Cumulative effort as modules are added. Illustrative shape, not a measurement.

What you would build before the first business module

- Multi-tenant isolation in two layers
- A tenant-aware identity broker
- A privilege and role model with org cascade
- A PII vault with tokenisation on every path
- A secret vault with envelope encryption
- A BPMN engine plus worklists
- A form runtime with a visual designer
- A universal data table
- A document and PDF generator
- A module registry and manifest schema
- A per-user menu pipeline and console shell
- A white-label theme system
- An AI gateway with audit
- An end-to-end tester and an integration gate

1

One platform to license, not fourteen to build.

A list this long is a programme of work, not a project. Zorbit is that programme, already done and proven end to end.

PRICING

Ask.

Platform licence, scoped to you. Write to hello@utunga.com.

hello@utunga.com

utunga

Zorbit is built and licensed by Utunga.

Zorbit, by Utunga · 2026