GREENWAYS

The App Platform That Grows Up

Business plan for AI-built applications that remain understandable at scale

Build the first version in a conversation. Build the hundredth feature without losing the system.

Category	AI application creation and lifecycle platform
Entry product	Prompt-to-working web application with database, auth, workflows and deployment
Wedge	Large applications whose behaviour remains legible, testable and safe to change
Initial buyer	Product studios, technical founders and internal application teams
Expansion	Multi-team governance, mobile clients, reusable domain packs and private deployment
Revenue	Subscription, build usage, runtime usage, enterprise control plane and implementation partners
Technical edge	Typed database behaviour, deterministic generation, headless application state and compiler-selected reasoning context

Working name and planning assumptions. Prices, conversion, costs and forecasts require validation through design partners and paid pilots.

The company in one sentence

Greenways is an AI application platform that gives a non-expert the immediacy of Base44 while giving a serious software team something current builders struggle to preserve as an application grows: a compact, executable and human-readable account of what the system does.

Contents

1 Executive plan

1.1 The opportunity

Conversational app builders have proved that software creation can start with intent rather than code. Base44 describes a product that creates frontend, backend, database and authentication from plain English, while Bolt, Lovable and Replit compete around the same compression of idea-to-deployment time [1, 6, 7]. Wix's acquisition of Base44 for approximately US$80 million plus performance earn-outs is strong strategic evidence that this is a commercially meaningful category [4].

The first generation of the category optimises the first hour. Greenways should optimise the first hour *and* the third year.

As an AI-built application becomes successful, it accumulates entities, permissions, workflows, integrations, background processes, views, mobile experiences and exceptions. General-purpose code generation expands the amount of active code and the context an agent must reconstruct. The customer can create faster, but each later change becomes less predictable. The product eventually crosses a complexity boundary at which prompting feels like negotiating with a system nobody fully understands.

Greenways can build a Base44-type experience on a different substrate. Instead of treating generated application files as the primary truth, it represents durable behaviour in a compact typed domain model, executes that behaviour directly in PostgreSQL, generates readable contracts and client state, and gives each AI change a compiler-selected reasoning packet. The user still describes the application conversationally. The platform quietly keeps it coherent.

Positioning

Other builders promise: "from idea to app in minutes." Greenways should promise: "from idea to an application your business can keep."

1.2 The initial business

The initial product is a hosted web application builder for data-rich operational software: customer portals, workflow systems, vertical SaaS, marketplaces, regulated administration, membership platforms and internal operations. These categories reward a fast first version but also accumulate substantial durable behaviour.

The initial customer is not every person with an idea. It is a small team that expects the app to matter:

- a product studio producing several client applications;
- a technical founder building a workflow-heavy SaaS product;
- an internal product or operations team replacing spreadsheets and fragmented tools;
- a domain consultancy that knows the workflow but lacks a large engineering team.

The platform uses a product-led entry and a sales-assisted expansion. A customer can build and publish a useful application without talking to Greenways. When the application acquires more users, data, risk or teams, the customer buys environments, governance, compatibility guarantees, private networking, audit evidence, mobile output and support.

1.3 Three-year objective

The objective is to establish a new decision criterion in AI app building: *maintainable complexity*. A credible three-year outcome is not market dominance. It is evidence that Greenways can repeatedly take applications beyond the point at which prompt-to-code systems become fragile.

	Year 1	Year 2	Year 3
Product proof	10 design-partner apps	Public self-serve platform	Repeatable scale tier
Paid accounts	150	1,000	4,000
Enterprise/org	6	30	100
Illustrative ARR	US$0.33m	US$3.0m	US$14.6m
Primary proof	Change safety	Retention and expansion	Efficient acquisition and governance

These are planning targets, not forecasts. Section 9 makes the assumptions explicit.

1.4 Why Greenways can attempt this

The repository already contains unusually deep enabling infrastructure: a Clojure-based multi-language compiler, typed PostgreSQL function definitions, deterministic RPC and client-contract generation, XTalk language targets, an event-driven substrate, portable headless application state, JavaScript worker execution and standalone Dart execution. These technologies are not the customer proposition. They reduce the cost of delivering the proposition.

The commercial task is to turn that capability into a narrow, delightful product loop, prove that its generated applications are easier to change, and avoid spending years generalising the compiler before customers are building applications.

The investment thesis

AI makes first-version code abundant. Durable understanding becomes scarce. A platform that can generate both the application and the bounded explanation required to change it can capture value after the novelty of instant generation has commoditised.

2 Customer and problem

2.1 The category has two different jobs

AI app builders are commonly discussed as if they perform one job. In practice they perform two:

1. **Creation:** turn an idea into a working application quickly.
2. **Evolution:** preserve the application's meaning while it changes repeatedly.

The first job is visible in a demo. The second becomes visible only after real users, real permissions, real data and real exceptions arrive. Greenways must be good enough at creation to enter the category and distinctly better at evolution to own a valuable segment.

2.2 The large-application failure pattern

A successful operational application does not merely acquire more screens. It acquires interacting rules. A subscription change affects billing, permissions, notifications, reporting, mobile state and historical records. A new role affects database policy, commands, visible actions, filters and audit evidence. A new client creates another interpretation of the same contract.

In a conventional AI-generated project, those decisions are distributed across schema files, migrations, API handlers, validation schemas, authorization middleware, frontend types, stores, components, tests and documentation. The problem is not that any one technology is poor. The problem is the number of places in which the same semantic decision may drift.

The customer experiences this as:

- prompts that fix one path and break another;
- rising token consumption as project files are repeatedly re-read;
- duplicated validation and authorization behaviour;
- types that compile while runtime meaning differs;
- reluctance to change mature workflows;
- rewrites when the prototype becomes the business;
- dependence on the original builder or an expensive rescue team.

2.3 Beachhead customer

The highest-value early customer has enough complexity to feel this pain, but can still adopt a new platform without a multi-year procurement process.

Segment	Trigger	Why it buys
Product studios	Several similar client builds; margin lost to maintenance	Reuse domain packs, reduce support burden, preserve handover quality
Vertical SaaS founders	Prototype is gaining customers and rules	Avoid a post-validation rewrite; add mobile and governance later
Internal app teams	Spreadsheet/process backlog with sensitive permissions	Ship quickly with central policy, audit and controlled publishing
Domain consultancies	Deep process knowledge but limited software capacity	Turn expertise into repeatable applications and packs

The recommended first vertical is **workflow-heavy B2B operations**, not generic consumer apps. Candidate examples include case management, member administration, approvals, service delivery, compliance evidence, field operations and multi-party marketplaces. These applications are database-centred, permission-sensitive and valuable before they require specialised graphics or extreme consumer scale.

2.4 The buyer, builder and user

The product must serve three roles without confusing them:

- The **builder** describes workflows, reviews generated behaviour and composes the application.
- The **technical approver** inspects permissions, migrations, tests, contracts and deployment evidence.
- The **application user** experiences ordinary, branded web or mobile software and need never know Greenways exists.

In a small company, one person may hold all three roles. In an enterprise, separating them becomes a governance feature and a revenue expansion path.

2.5 The outcome customers purchase

Customers are not purchasing generated lines of code. They are purchasing compressed delivery risk:

1. prove a workflow against a real database immediately;
2. publish a credible first application quickly;
3. add features without multiplying independent truths;
4. understand what a proposed change will affect;
5. retain ownership of data, contracts and readable generated artifacts;
6. bring in professional developers without abandoning the platform.

The primary success metric should therefore be **time to safe change**, not time to first render.

3 The product

3.1 The product loop

The customer experience should feel conversational, visual and immediate. The internal process can be rigorous without exposing compiler terminology.

1. **Describe.** The builder explains the business, actors, records, permissions and desired workflow in ordinary language.
2. **Structure.** Greenways proposes a visible application map: domains, entities, roles, states, commands, views, events and integrations.
3. **Prove.** The platform creates an isolated PostgreSQL environment, executes the durable behaviour and runs generated examples and permission tests.
4. **Compose.** It generates headless application models and an editable web interface using a selected design system.
5. **Preview.** The builder interacts with seeded scenarios, including failure, empty, pending and unauthorized states.
6. **Publish.** Greenways deploys the application, database changes and signed build evidence through controlled environments.
7. **Evolve.** Each later prompt becomes a proposed semantic change with impact, migration, tests and preview before application.

The decisive interface is not the chat box. It is the **application map** that accumulates a stable, navigable model of the system.

3.2 The application map

Large applications need a spatial memory. The application map should let a builder move between:

- business domains and functional slices;
- data records and lifecycle states;
- roles, permissions and policy explanations;
- commands, reads, events and external integrations;
- pages, mobile screens and headless state models;
- tests, example scenarios and production evidence;
- change history and compatibility boundaries.

Every item should answer four questions: What does it mean? Where is its authoritative definition? What consumes it? What proves it works?

3.3 The safe-change experience

A later prompt such as “allow regional managers to approve refunds below $2,000” should not immediately rewrite files. The platform should display:

- the proposed new permission and threshold;
- affected database functions, policies, events and views;
- the web and mobile actions that become available;
- migration and backward-compatibility consequences;
- generated examples for allowed and denied cases;
- a runnable preview with representative roles;
- the exact source and generated artifact diff.

The builder may accept, refine or reject the change. This is a stronger product primitive than unconstrained code editing because it preserves intent before implementation.

3.4 Product editions

Edition	User	Product boundary	Expansion trigger
Explore	Individual	Public/sandbox apps, core builder, limited usage	Private app or custom domain
Builder	Founder/studio	Production apps, Git sync/export, environments, packs	Collaboration and multiple applications
Team	Product team	Shared workspace, review gates, reusable modules, usage pool	Governance, network and assurance
Scale	Growing business	Higher runtime limits, mobile output, audit history, support	Procurement and private infrastructure
Enterprise	Organization	SSO/SCIM, policy control, private runners, SLAs, data residency	Portfolio-wide adoption

3.5 What the product should not do initially

The first release should not promise arbitrary software. It should constrain the problem deliberately:

- responsive database-backed web applications first;
- a small number of approved UI systems;
- PostgreSQL as the durable system of record;
- supported workflow, messaging, file, payment and AI integration patterns;
- generated mobile only after web and headless-state parity is proven;
- external workers for long-running, CPU-heavy or integration orchestration tasks.

Constraints are part of the reliability proposition. An escape hatch may exist for professional developers, but unsupported code must be visibly outside the platform's compatibility guarantee.

The emotional promise

The customer should feel that the application becomes more knowable as Greenways builds it. The product is not a black box that occasionally reveals code. It is a living explanation that also runs.

4 Why Greenways wins on large applications

4.1 Complexity without collapse

Application complexity is not inherently bad. A large application may represent a valuable organization faithfully. Collapse occurs when the cost of understanding interactions rises faster than the value of adding behaviour.

Greenways should make complexity additive rather than combinatorial. Each functional slice owns durable behaviour, generated contracts, headless state and evidence. Slices interact through explicit commands, views and events. A new feature expands the application map, but should not require an agent to reinterpret the whole repository.

4.2 The reasoning packet

For every proposed change, the compiler graph can select a bounded packet containing:

- the authoritative domain definitions;

- relevant schema, permissions and lifecycle rules;
- neighbouring commands, views and events;
- generated target contracts and source maps;
- focused tests and representative scenarios;
- compatibility history and active consumers.

The AI agent receives that packet rather than a broad, heuristic sample of the codebase. The human reviewer receives the same packet in a readable form. This aligns AI context, review context and executable evidence.

If a five-million-token application repository can be reduced to a complete fifty-thousand-token reasoning packet, the context reduction is:

$$K_{context} = \frac{5{,}000{,}000}{50{,}000} = 100\times$$

The ratio is a testable design target, not a universal claim. A packet that omits a necessary dependency has failed even if it is small. The commercial measure is the smallest *complete* context that produces a correct change.

4.3 Types are necessary but insufficient

Type generation helps when it removes repeated data-shape declarations. It does not by itself make a large application coherent. TypeScript types disappear at runtime, and even runtime schemas do not necessarily unify permissions, transactions, errors, events, migrations and client behaviour.

Greenways begins with executable database behaviour. PostgreSQL functions define durable state transitions close to the data and can be tested immediately. The compiler then derives ordinary SQL, RPC metadata, target-language contracts and headless state. The advantage is not simply “more types.” It is fewer independently maintained meanings.

4.4 Readable output is a product requirement

Generated code must be legible because customers will eventually need to inspect, audit, extend or leave the platform. Greenways should enforce:

- stable naming across database, contract, state and UI layers;
- deterministic formatting and generation;
- source maps from every generated artifact to its owner;
- explicit commands, permissions, errors and events;
- semantic diffs in addition to file diffs;
- no hidden production behaviour that exists only inside an AI conversation.

The platform’s export story should include code, data, schema, migrations, tests, contracts and provenance. Exporting only frontend files is not sufficient for a serious application.

4.5 How the advantage compounds

As the app grows	Generic prompt-to-code pressure	Greenways design response
More entities	More schema, API and form synchronization	Typed selectors and returns generated from the database owner
More roles	Policy duplicated across layers	Durable permission functions and generated capability surfaces
More workflows	State transitions scattered through handlers and UI	Explicit commands, events and headless models
More clients	Each client interprets contracts independently	Shared identifiers and generated JS/Dart contracts
More history	Agents repeatedly reconstruct old decisions	Provenance, compatibility history and reasoning packets
More builders	Inconsistent conventions and hidden assumptions	Compiler constraints, packs, review gates and semantic diffs

This is the product’s retention engine. The more legitimate complexity the customer accumulates, the more valuable the application map and verified history become. Lock-in should come from accumulated understanding and operational confidence, not from preventing export.

5 Market and competition

5.1 A validated and crowded category

The category includes fully hosted no-code builders, code-generating builders, browser development environments and AI coding agents. Base44 now combines conversational generation with an integrated backend, hosting, database, authentication and enterprise controls. It also advertises code export and two-way GitHub sync on paid plans [2, 3]. Bolt explicitly acknowledges that larger projects consume more AI tokens because their file systems must be synchronized into model context [5]. Replit combines an autonomous agent, development environment, database and deployment [7].

Greenways should not claim that competitors cannot build production applications or export code. Their products are advancing quickly. The defensible claim is narrower: Greenways is designed from the beginning around preserving semantic coherence across large, long-lived, multi-client applications.

5.2 Competitive map

Category	Representative products	Primary strength	Greenways opening
Integrated AI builder	Base44	Complete no-code app and hosted services	Verifiable domain behaviour and large-app reasoning
Code-generating builder	Lovable, Bolt, v0	Fast visual iteration and familiar web code	Bounded context, durable semantics, cross-client contracts
Cloud IDE agent	Replit	Flexible agent, runtime and deployment	Opinionated application architecture and change proof
Developer agent	Codex, Claude Code, Cursor	Works across existing stacks	Product experience for domain builders and governed portfolios
Traditional low-code	Retool, Mendix, OutSystems	Enterprise governance and integrations	Conversational creation, readable ownership and lower implementation friction
Internal framework	In-house platform teams	Exact organizational fit	Productized compiler and lower fixed platform cost

5.3 The category wedge

Greenways should avoid a feature-by-feature contest on landing pages, novelty applications or unrestricted framework choice. Its wedge is the point where an application becomes operationally important:

- more than one role and permission boundary;
- several interacting domains or workflows;
- audit, migration or data-integrity requirements;
- a second client, such as mobile or an external API;
- several builders or teams making changes;
- a long expected application life.

The competitive demonstration should start with the same medium-sized specification in several builders, then add ten consequential changes. The winning demo is not the prettiest first screen. It is the platform that can explain and implement change ten without regressing changes one through nine.

5.4 Bottom-up market model

Top-down software market figures are too broad to guide an early company. The plan uses a bottom-up serviceable-account model that must be validated.

Reachable segment assumption	Accounts	Target annual value	Illustrative serviceable revenue
Product studios and agencies	20,000	US$6,000	US$120m
Vertical SaaS and technical founders	50,000	US$3,000	US$150m
Internal application teams	15,000	US$18,000	US$270m
Enterprise portfolios	2,000	US$75,000	US$150m
Illustrative serviceable market	87,000	–	US$690m

These are scenario inputs, not sourced counts. The first customer-discovery programme must replace them with observed segment sizes, procurement patterns and willingness to pay. The near-term obtainable market is deliberately smaller: 4,000 self-serve accounts and 100 organizational accounts, corresponding to the Year 3 plan.

5.5 Strategic evidence

Base44's acquisition validates buyer interest and strategic option value, but it does not validate Greenways' architecture or market entry. The transaction also means a direct clone would compete against Wix distribution. Greenways therefore needs a distinct proof: customers choose it because their applications continue to grow safely, not because it offers another chat box.

6 Business model

6.1 Revenue architecture

The platform should charge for four forms of value separately enough to preserve healthy economics:

1. **Creation capacity:** AI planning, generation, repair and design operations.
2. **Application operation:** database, storage, events, workers, bandwidth and observability.
3. **Organizational control:** collaboration, review, audit, policy, identity and private infrastructure.
4. **Reusable capability:** supported domain packs, connectors and partner-delivered implementations.

A single opaque credit obscures the difference between expensive model work and predictable runtime consumption. Greenways can present a simple allowance to small customers while preserving separate metering internally.

6.2 Illustrative pricing

Plan	Price	Includes	Commercial purpose
Explore	Free	One sandbox app, limited builds and runtime	Product discovery and template sharing
Builder	US$49/mo	Private production app, export, custom domain, environments	Founder conversion
Studio	US$199/mo	Five builders, multiple apps, shared packs, pooled build usage	Agency and small-team retention
Scale	US$799/mo	Higher runtime, mobile target, audit, advanced environments and support	Growing operational applications
Enterprise	US$30k–150k/yr	SSO/SCIM, private runners, policy, network, residency and SLA	Governance and portfolio expansion

Model and runtime overages should be sold at a transparent margin. Domain packs can be included, rented or purchased. A marketplace can later share net revenue with verified partners, but should not precede repeated internal pack use.

6.3 Expansion mechanics

Revenue should expand with legitimate customer success:

- more production applications and environments;
- more builders, reviewers and viewers;
- increasing runtime and integration activity;
- mobile and external API consumers;

- longer evidence retention and stricter compatibility policy;
- reusable private packs across an application portfolio;
- support, migration and private deployment requirements.

The platform should not make applications artificially expensive to leave. Complete export is commercially compatible with retention if the hosted product continues to provide valuable reasoning, verification, operation and governance.

6.4 Unit-economics model

The contribution margin for account i is:

$$CM_i = S_i + U_i + E_i - (M_i + I_i + H_i + P_i)$$

where S is subscription revenue, U usage revenue, E enterprise or pack revenue, M model cost, I infrastructure, H variable support and P payment or partner cost.

The model should target:

- self-serve gross margin above 70% after model and hosting cost;
- enterprise gross margin above 75% after allocated support;
- blended net revenue retention above 115% once expansion cohorts mature;
- self-serve CAC payback below six months;
- sales-assisted CAC payback below eighteen months;
- model cost below 15% of self-serve revenue through bounded context and task routing.

The context architecture is economically important. If a mature app requires fewer input tokens per safe change, Greenways can offer better margins, more predictable allowances or both. The 100-fold technical hypothesis should be tested as a cost curve: median model input per accepted change versus application size.

6.5 Services without becoming an agency

Paid design-partner implementations are useful in the first year because they expose missing product capability and create reference applications. Services should obey three rules:

1. each engagement must produce reusable platform capability or evidence;
2. custom work must use the same public product path intended for customers;
3. recurring software revenue must become the majority of account value after implementation.

Certified partners should eventually deliver migrations and vertical solutions. Greenways retains the platform relationship, verification standard and usage revenue.

7 Go-to-market

7.1 Start with consequential applications

The launch message should not be "build anything." It should show a specific class of application becoming credible quickly and remaining coherent through repeated change.

The recommended entry offer is a **ten-day application proof** for product studios and operational teams:

1. select one workflow with real roles, records and edge cases;
2. build a working application in the customer's environment;
3. add three consequential change requests after the first version;
4. deliver the application map, executable tests and export package;
5. compare time, defects and explanation effort with the customer's current approach.

The proof should be paid where possible. A price of US$5,000–15,000 is low relative to a custom build but high enough to identify real demand. Fees may convert into platform credit for customers who proceed.

7.2 Design-partner programme

Recruit ten design partners across three repeatable patterns rather than ten unrelated applications:

- approval and case-management workflows;

- member, customer or supplier portals;
- multi-party operational marketplaces.

Each partner must agree to provide anonymized benchmark data and participate in structured change exercises. The programme succeeds when at least six applications remain active, four customers pay recurring fees and two patterns produce reusable packs.

7.3 Product-led acquisition

The free experience should demonstrate the product's difference, not merely ration prompts. Useful acquisition loops include:

- public application-map templates that can be forked;
- "change challenge" demos showing ten safe iterations;
- import of an existing schema or OpenAPI contract to produce a legibility assessment;
- shareable build evidence and architecture reports;
- domain-pack galleries with working seeded applications;
- transparent comparisons of token consumption as applications grow.

Search content should target high-intent problems: replacing an internal spreadsheet, building a member portal, escaping a fragile prototype, adding mobile to a workflow application, or auditing an AI-generated codebase.

7.4 Studio and consultancy channel

Studios are both customers and distribution. A studio can use Greenways to improve margin, sell ongoing platform assurance and hand over a more legible application. The partner programme should provide:

- multi-client workspaces and white-label presentation;
- reusable private packs and design systems;
- certification in application modelling and safe extension;
- referral or recurring revenue share;
- escalation support and compatibility guarantees;
- migration tools for existing client applications.

The channel becomes defensible when partners encode domain expertise as transparent packs that attract more customers and improve the core compiler's evidence corpus.

7.5 Enterprise motion

Enterprise sales should follow observed application adoption rather than lead the company. The trigger is an organization with several successful team-built apps asking for central control. The enterprise offer then solves portfolio questions:

- who may build, review, publish and access applications;
- which connectors, models, data locations and packs are approved;
- which contracts changed and which applications consume them;
- whether every production release has executable evidence;
- how applications can run in private networks or controlled regions.

Land with a workflow; expand through the application portfolio.

7.6 Launch proof, not launch theatre

The most persuasive launch asset is a public reference application with at least fifty meaningful functions, multiple roles, web and mobile clients, seeded failures, an evolving migration history and a sequence of recorded change requests. Every layer should be inspectable. This demonstrates the product's claim more credibly than a collection of one-prompt demos.

8 Technology and defensibility

8.1 The enabling stack

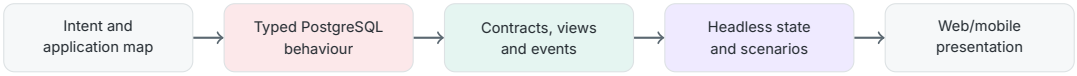
Greenways has several technologies that can make application construction easier to reason about:

Capability	Product effect	Commercial effect
Clojure source and macros	Compact, composable domain declarations	More behaviour per maintained source unit
Typed PostgreSQL functions	Durable behaviour executes beside data	Immediate proof, consistent transactions and permissions
Readable SQL emission	Ordinary target artifacts can be inspected	Trust, exportability and easier technical approval
Generated RPC/contracts	Clients consume one semantic owner	Lower drift and cheaper multi-client support
XTalk language targets	Equivalent JS, Dart and other target interfaces	Web/mobile expansion without parallel reinvention
`xt.substrate`	Event, cache, source and synchronization backbone	Offline/mobile/runtime product packs
`xt.ui` **headless state**	Application workflows independent of UI framework	Test before presentation and support native composition
Deterministic build graph	Provenance and repeatable artifacts	Verified builds, compatibility policy and enterprise control

The builder should never need to learn Clojure or PL/pgSQL. The product translates intent into a reviewed domain model and renders explanations in business language. Professional developers can inspect and extend the compact source when necessary.

8.2 Architecture of one functional slice

The supported sequence is:



The platform tests each boundary and keeps presentation replaceable. Database-first does not mean database-only: long-running work, third-party integrations and compute-heavy tasks belong in typed workers, while user experience belongs in native web or mobile composition.

8.3 The moat is accumulated proof

A DSL alone is copyable. Defensibility accumulates through:

- a large corpus of paired intent, domain models, readable SQL, tests and accepted changes;
- inference about which dependencies form a complete reasoning packet;
- compatibility fixtures across PostgreSQL, JavaScript, Dart and supported runtimes;
- domain packs proven in real operational applications;
- migration knowledge from conventional and AI-generated projects;
- partner expertise and customer-specific application maps;
- trust in deterministic, exportable and reproducible builds.

The data advantage should come from product operation with explicit customer controls, not from secretly training on private application content. Anonymous structural metrics and opt-in examples can improve routing and benchmark quality without appropriating customer IP.

8.4 Open versus proprietary boundary

Open or source-available components can increase trust and adoption:

- the application specification format;
- generated contracts and target runtimes;
- local build and test tooling;
- export, source maps and conformance fixtures.

Commercial value can remain in the collaborative builder, AI planning, hosted environments, reasoning graph, verified build service, governance control plane, private runners, managed upgrades and supported packs.

This boundary makes exit credible while preserving a strong hosted product.

9 Operating and financial plan

9.1 Product phases

Phase	Period	Product outcome	Commercial outcome
Proof	0–6 months	One polished workflow builder; deployable web app; application map; safe changes	Three paid design partners
Private beta	6–12 months	Auth, files, notifications, environments, export, usage metering	Ten partners; 150 paid accounts; six org contracts
Public product	12–24 months	Self-serve onboarding, packs, Git workflow, observability, partner workspace	1,000 paid accounts; 30 org contracts
Scale	24–36 months	Mobile target, governance control plane, private runner, migration tooling	4,000 paid accounts; 100 org contracts

Each phase is gated by repeated customer behaviour, not completion of the underlying technology catalogue.

9.2 Three-year planning model

The model separates self-serve subscription revenue and organizational annual contracts.

US$ millions unless stated	Year 1	Year 2	Year 3
Average paid self-serve accounts	150	1,000	4,000
Average self-serve MRR/account	$100	$150	$180
Self-serve ARR run-rate	0.18	1.80	8.64
Organizational contracts	6	30	100
Average organizational ACV	$25k	$40k	$60k
Organizational ARR run-rate	0.15	1.20	6.00
Total ARR run-rate	0.33	3.00	14.64
Gross margin assumption	68%	74%	78%
Gross profit run-rate	0.22	2.22	11.42

The account counts are end-state planning targets while average MRR and ACV are mix assumptions. Recognized revenue will lag ARR during growth. A cash model must therefore use monthly cohorts, billing timing and actual usage costs before fundraising.

9.3 Illustrative operating envelope

US$ millions	Year 1	Year 2	Year 3
Recognized revenue assumption	0.18	1.65	8.20
Cost of revenue	0.07	0.43	1.80
Product and engineering	1.20	2.10	3.80
Sales, success and partnerships	0.30	0.85	2.20
Operations, security and admin	0.25	0.45	0.90
Illustrative operating result	-1.64	-2.18	-0.50

This envelope suggests approximately US$4.5–5.5 million of external capital or founder-funded equivalent to reach a Year 3 break-even trajectory with contingency. A leaner services-supported path could reduce the requirement but may slow the self-serve product.

9.4 Team plan

The first twelve months require a small cross-functional team:

- founder/product and architecture lead;
- two compiler/database/runtime engineers;
- two product engineers spanning builder, web presentation and deployment;
- one product designer with strong information-architecture capability;
- one design-partner engineer/customer success lead;
- fractional security, finance and legal support.

The key hiring risk is over-concentrating on compiler depth while underinvesting in onboarding, visual composition and the emotional quality of the builder. The product must feel easier than conventional development even though its internals are more disciplined.

9.5 Fundraising structure

A credible seed narrative funds eighteen to twenty-four months to prove:

1. customers can build a useful first application without the core team;
2. applications survive consequential changes with lower failure and explanation cost;
3. at least one customer segment retains and expands at attractive margins;
4. reasoning context grows sublinearly with application size;
5. partners can deliver applications through the product rather than bespoke engineering.

Capital should accelerate a working product loop and distribution. It should not fund an open-ended attempt to support every programming language, database or application category.

10 Risks, evidence and decision gates

10.1 Principal risks

Risk	Consequence	Mitigation and evidence
Creation experience is too technical	Users choose easier builders despite later fragility	Test non-expert onboarding; hide compiler concepts behind the application map
Competitors improve large-project context	Differentiation narrows	Benchmark semantic consistency, not token window size; compound domain and compatibility evidence
PostgreSQL boundary is too restrictive	Important workloads require escape paths	Define workers and integrations explicitly; target database-centred workflows first
Generated UI is mediocre	Product cannot acquire users	Invest early in design systems, native override and excellent default compositions
Central compiler defect	Many applications receive the same error	Versioned IR, conformance suites, staged rollout, rollback and signed provenance
Platform appears proprietary	Technical buyers fear lock-in	Complete export, readable artifacts, local tests and documented runtime boundary
Model costs erode margin	Usage grows faster than revenue	Bounded reasoning packets, model routing, caching and transparent overages
Services consume the company	Roadmap fragments into client work	Accept only engagements producing repeated product capability
Security event damages trust	Enterprise adoption stalls	Least privilege, isolation, external testing, secure SDLC and narrow early scope

10.2 The benchmark programme

Greenways needs evidence that matches its claim. Select three substantial functional slices and implement them in Greenways and strong conventional stacks. After the first version, issue a fixed sequence of changes involving permissions, migrations, events, mobile state and failure recovery.

Measure:

- time to first working application;
- time to each accepted change;
- regressions and hidden-edge-case failures;
- tokens and files inspected per change;
- time for an unfamiliar engineer to explain the behaviour;
- number of authoritative semantic surfaces;
- source-to-artifact trace coverage;
- change success after six and twelve months.

The 100-fold context target passes only when the reduced packet is complete enough to achieve equal or better correctness. The large-app thesis passes only when advantage increases, or at least does not decay, as the application grows.

10.3 Twelve-month milestones

1. Publish one inspectable reference application and its first twenty change episodes.

2. Build the conversational intent-to-application-map workflow.
3. Support a complete production slice: data, auth, roles, files, notifications, web UI, environments and export.
4. Generate semantic impact previews and focused reasoning packets.
5. Recruit ten design partners in no more than three workflow families.
6. Convert at least four design partners to recurring subscriptions.
7. Demonstrate three customer-created applications without core-team implementation.
8. Measure gross margin and context cost at three application sizes.
9. Complete an external security review before handling high-risk production data.
10. Decide whether mobile generation is the next expansion based on customer pull.

10.4 Decision gates

Gate	Evidence required	Decision
First-app parity	Non-experts reach a working application at category-competitive speed	Continue product-led builder investment
Large-app advantage	Safe-change benchmark improves materially after ten changes	Lead with maintainable complexity
Paid urgency	Four of ten design partners convert and expand usage	Invest in the winning segment
Self-serve repeatability	Majority of beta apps launch without core-team engineering	Open public acquisition
Economic advantage	Model plus infrastructure cost remains below target as apps grow	Offer generous usage and scale acquisition
Partner repeatability	Two partners deliver applications using reusable packs	Build certification and marketplace
Enterprise pull	Several customers request portfolio governance after team adoption	Build control plane and private runners

11 Conclusion

Base44 and its peers have established the new minimum expectation: a person should be able to describe an application and see it working. Greenways should meet that expectation, then extend it to the part of the lifecycle where businesses incur most of their risk.

The differentiated product is not Clojure, PostgreSQL, generation or types. It is an application that carries its own compact explanation: authoritative behaviour, readable artifacts, explicit impact, executable evidence and bounded context for the next change.

That creates a simple category position:

> **The promise**
>
> Build as quickly as a modern AI app builder. Keep building when the application becomes large, valuable and complicated.

The immediate company task is narrower than the long-term platform vision. Choose one workflow-rich beachhead, make the creation experience delightful, prove twenty consequential changes in public, and charge design partners for applications they intend to keep. If Greenways can demonstrate that understanding scales with the product, it has the basis for a better app-building company rather than merely a better build system.

References

[1] Base44, *Enterprise app development for every team*, product materials current 20 July 2026. https://base44.com/enterprise.

[2] Base44, *Plans and Pricing*, current 20 July 2026. https://base44.com/pricing.

[3] Base44, *GitHub Integration*, support documentation current 20 July 2026. https://docs.base44.com/developers/app-code/local-development/github.

[4] Wix, *Wix Further Expands into Vibe Coding with Acquisition of Base44*, 18 June 2025. https://www.wix.com/press-room/home/post/wix-further-expands-into-vibe-coding-with-acquisition-of-base44-a-hyper-growth-startup-that-simplif.

[5] Bolt, *Plans and pricing*, current 20 July 2026. https://bolt.new/pricing.

[6] Bolt, *AI app builder*, current 20 July 2026. https://bolt.new/use-cases/ai-app-builder.

[7] Replit, *Agent and Pricing*, current 20 July 2026. https://replit.com/products/agent; https://replit.com/pricing.

[8] Lovable, *AI app builder product and pricing materials*, current 20 July 2026. https://lovable.dev.

[9] Greenways, *xt.substrate Functional Slice Architecture*, repository architecture guide 0020, 2026.

[10] Greenways, *xt.ui Application Boundary*, repository architecture guide 0030, 2026.

[11] Greenways, *The Legible Build*, technical and commercial working paper, 2026.