ballastlane
APPLICATIONS

CAPABILITY OVERVIEW

The BLA Software Factory

A structured, AI-enabled software delivery system built and run by Ballast Lane Applications, under the direction of a product owner, senior engineer, and you.

Introduction

Ballast Lane Applications (BLA) continues to make significant investments in AI training and tooling including certifications for all our engineers, the development of libraries, and cataloging of best practices for AI-enabled software development. Based on the learnings from these investments, plus our work on many cutting-edge projects, we have created our own Software Factory that is available for project use.

It is more than a reference implementation. It is a structured setup for building software applications under the direction of a product owner and senior engineer, designed to overcome the weaknesses found in other AI-enabled approaches such as bias, blind spots, hallucinations, and suspect foundational decisions.

We have already used it with great success for several engagements, including:

- PoC / MVP development for new feature exploration
- Migration of an application to a full production environment
- Creating new features on an existing code base
- Estimating level of effort and finding potential roadblocks for large-scale initiatives

For all its capabilities, the Factory still relies on input and guidance from an experienced team to ensure that the right features are built, and that the architecture and details meet the requirements for performance, maintainability, and security.

This document describes what the Software Factory is, what it delivers, how you interact with it, and what you receive at the end of an engagement.

Software Factory Definition

The Software Factory is a software delivery system run by BLA and can take on a variety of tasks, including:

- New software products from a written specification
- Individual features or modules added to an existing codebase
- Prototypes hardened to production quality
- Platform migrations and compliance rewrites
- Multi-module integrations between systems

A minimum of two staff members are assigned to every engagement with the Factory: an Operator (a senior engineer) and a Planner / QA coordinator (usually a Senior Product Owner). The Operator runs the agent pipeline, handles hard technical problems such as edge cases, integrations, and security, and provides final approval on all architecture decisions. The Planner turns your priorities into scoped work, runs independent verification and tests, manages tradeoffs, and is the primary point of contact. Additional team members may be required depending on scope, complexity and customer support requirements.

The Factory itself is a structured system with defined roles, checkpoints, and documented outputs, coordinating a fleet of AI agents that draw from multiple AI providers to write, review, test, and ship software. The AI does the execution work under the supervision and guidance of the BLA team and the client.

The agent fleet has two layers. First, the Orchestrator layer, which reads each task, decides which specialist should handle it, dispatches the work, and assembles the results. It is the equivalent of a project manager for the AI team.

Second, the Specialist layer, where each specialist is configured for one type of task, has access only to the tools that task requires, and can be modified to handle work specific to your project.

- **Engine developer:** Writes core product logic
- **Frontend developer:** Builds client-facing interface
- **Test author:** Writes automated tests
- **Reviewer:** Checks code for bugs and standards
- **Architect:** Approves or blocks structural decisions
- **Coordinator:** Handles work spanning multiple areas

Three rules govern the fleet:

1. Reviewers and Architects cannot change code, so no single agent can both produce and approve its own work.
2. Only the Orchestrator dispatches work — agents cannot assign tasks to each other, which avoids looping and wasted tokens.
3. Specialists run in parallel, with multiple workstreams proceeding simultaneously to ensure rapid execution.

For significant decisions — including architecture choices, security-sensitive implementations, and major structural calls — the Factory uses a process called Council Mode. A position is put forward, then a critic agent from a different AI provider family attacks it. The original agent rebuts, and the result is re-evaluated before a human gate.

> *The governing rule: no single agent can lock in a load-bearing decision.*

The AI providers used in the fleet include Claude (Anthropic), GPT (OpenAI), Qwen (Alibaba), DeepSeek, and others. Different providers are used for different roles — orchestration, bulk execution, and independent critique — so that no single vendor's blind spots affect your delivery. This also means the Factory is not dependent on any

one provider with the exception of the Orchestrator which depends on Anthropic because of its long horizon context management. If a provider has an outage or a price change, work routes around it automatically.

Where Work Runs

Each engagement runs in its own isolated container on Ballast Lane machines. Engagements are not on a shared public cloud, and client credentials are kept separately. AI agents do not hold standing connections into your systems; access is granted only to short-lived workers that are destroyed when a task finishes.

Data and AI Training

In most cases, the commercial AI services used by the Factory operate under API terms that do not use your inputs or outputs to train their models. We can exclude the few who do not if your security policy requires. Every model call passes through a single recorded layer, which means we can show you exactly what model touched your data, when, and what was changed. The full data posture is set per engagement and governed by contract before any work begins.

Private Model Option

For sensitive engagements — such as work involving protected health information — the Factory can run key data handling activities on self-hosted models on BLA hardware. The client's code and data never leave the machine and are never sent to any third-party AI provider.

We currently operate a validated local inference stack on Apple Silicon hardware running open-weights models. The specific models used are evaluated against a standard six-test suite before inclusion and are updated as stronger options become available.

Existing Code Bases

The Factory works on existing codebases as well as greenfield builds. For an existing codebase, there is a ramp-up period while the team maps what is already there — understanding the product, features, architecture, the conventions, and any known issues — before shipping begins. The decision log from that ramp-up becomes part of your permanent record for the engagement.

What the Factory Delivers

Every engagement produces working, production-ready software committed to your repositories. This section describes what is built and how delivery is structured across application code, infrastructure, security, and testing.

Application code

The Factory builds full-stack software. The standard technology mix is React / Next.js on the frontend, Node.js or Python on the backend, and TypeScript throughout. Examples of what has been delivered: multi-tenant SaaS platforms with role-based access control, care-planning and case-management workflows, real-time messaging, billing integrations (e.g. Stripe), REST and GraphQL APIs, authentication and authorization systems, and data import / export pipelines.

Work is broken into numbered iterations with discrete, testable units of functionality. Each iteration has a clear scope. Every item in your contract is mapped to the iteration where it was delivered, and that mapping appears in the scope coverage report in your verification packet.

Infrastructure

The Factory sets up and configures the full hosting and delivery environment as part of every engagement. What is delivered:

- CI/CD pipeline — automated build, test, and deploy triggered on code change
- Cloud hosting configuration — environment setup, variables, and secrets management

- Deployment scripts and container configuration
- Monitoring hooks and structured logging configuration
- Database provisioning and migration scripts
- Scaling configuration appropriate to the application

All infrastructure decisions and configuration choices are documented in the runbook delivered with every engagement.

Security

Security review runs on every iteration, not just at the end of an engagement. It has two parts.

- **Automated dependency scanning:** All third-party libraries are checked against known vulnerability databases. Vulnerable dependencies are flagged and patched before each delivery.
- **Code-level security review:** Security-sensitive code — including authentication flows, authorization checks, data handling, input validation, and API boundaries — is reviewed by an AI model from a different provider family than the one that wrote it. Findings are documented in the quality evidence file in the verification packet.

By the time of delivery, the codebase has no known vulnerable dependencies, and all security-sensitive code paths have been independently reviewed and documented.

Automated testing

A test suite is written alongside the application code as part of every engagement. The Test Author specialist is dedicated to this — tests are not added at the end. Tests cover:

- Unit tests for core business logic
- Integration tests for API endpoints and database interactions
- End-to-end tests for critical user flows

All tests run automatically in the CI/CD pipeline on every commit. A feature or fix is not marked delivered until tests pass.

Documentation

Every Factory delivery includes a verification packet: a structured ZIP archive documenting what was built and proving that it meets the agreed scope. It contains the technical deliverables, quality evidence, decision log, run book and other key materials.

How You Interact with the Factory

You communicate with the Factory through a dedicated Slack channel (e.g. **#yourcompany-ballastlane**). There is no portal, no ticketing system, and no standups to attend. You write requests in plain language that define features, changes, and questions for the Factory to pick up.

How requests work

- You send a request in the Slack channel: a new feature, a bug fix, a change to existing behavior.
- A bot watches the channel continuously and picks up the message — even replies on old threads.
- The request becomes a tracked item. You are notified on every update, so nothing is missed.
- The Factory works the request and replies with status, output, and any questions that need your input.
- Working software lands in your repository when the item is done, with a note on what changed.

The Factory handles one request at a time and only marks it done once the work has succeeded. If something goes wrong, the request stays on the list and is retried. Nothing is silently dropped.

When the Factory needs your input

The Factory comes to you only for decisions that require your judgment: product trade-offs, design choices, priority calls, and gate approvals. Everything else is handled without you. In essence, you provide direction on what to build and why, on product and design decisions, and on gate approvals.

What you own

At the end of any engagement, you own all code in your repositories, the verification packet, the runbook, and the decision log. There is no key-person dependency on BLA staff. Any competent engineering team can pick up the codebase from the documentation provided.

Why — and How — to Engage with BLA

Any team can now reach for AI to build software. Almost none can do it without inheriting the failure modes: hallucinated architecture, plausible code no one can vouch for, security decided by whoever happened to write the prompt. The Factory closes that gap: the speed of a fleet of AI agents, with the judgment, review, and evidence of a senior engineering team standing behind every decision.

What the Factory gives you is not a discount on code. It is the full output of an engineering team: planning, building, testing, security, and documentation. All running at AI speed, under people who can be held accountable for it.

Where you start depends on what you have today. You can pick one entry point or evolve from one to the next.

Prove it on a real project

> *For any team that wants to see the capability before committing to it.*

One delivery cycle, led by us, on a real product of yours, taken to a real, evidenced outcome. The lowest-risk way to watch the Factory prove itself — you finish with working software and a verification packet in hand, not a pitch deck.

We become your engineering team

> *For a founder or company that needs to ship but has no engineering organization — and doesn't want to spend a year building and managing one.*

We run as your delivery engine. You set direction and priorities; the Factory plans, builds, tests, secures, and ships against your roadmap, continuously. You get the output of a full-function team with senior judgment on every architectural call and a verification packet behind every release for roughly the cost of two senior hires. It is the team you couldn't realistically hire or manage, operating from day one.

Level up your own team

> *For an engineering organization that's fallen behind on AI and needs to catch up without betting the company on it.*

Buying tools does not change how a team works, and training decks do not stick. We embed AI-enabled product owners, designers, engineers, and builders directly into your teams, where they ship real work and transfer the practice by example. Your people learn by building alongside someone who already does it well, and the capability stays after we leave.

Own your factory

> *For a capable engineering organization that wants to industrialize AI development in-house and run it itself.*

Building a governed, multi-agent factory from scratch is months of expensive trial and error. Most attempts fail on the governance, the exact problem the Factory already solves. We set up a factory tuned to your stack and standards, transfer the operating model to your people, and hand you the keys. You get the blueprint and the hard-won lessons instead of paying to rediscover them.

Start a conversation

Tell us what you want to start building.

Email: support@ballastlane.com
Web: ballastlane.com
Phone: (781) 631-2536
Office: 1 Spring Street, Suite 1, Marblehead, MA 01945