Suhas Bhairav

AI Legacy Code Migration Refactoring Guide

A pragmatic, AI-assisted plan to modernize and secure legacy engineering code.

Pragmatic, AI-assisted migration plan to safely modernize code in weeks.

Suhas Bhairav

suhasbhairav.com

Automate, Review, and Tools

This page defines what to automate, what to keep human-reviewed, and the data and tooling needed to start safely.

Key move **Start small: pick a non-critical module to pilot the AI-assisted refactor.**

PRACTICAL CHECKLIST

01	Inventory critical legacy modules and their business impact to prioritize first migrations.
02	Automate common refactors: syntax upgrades, dependencies, and simple API changes using AI tooling.
03	Keep core business logic and security decisions under human review and approvals.
04	Capture data: repo history, test coverage, dependencies, and runtime metrics.
05	Tools: AI coding assistants, static analyzers, test-generation, and CI integration.
06	Example: migrate Python 2 module to Python 3 with typing and tests.
07	Example: upgrade legacy Java components to modern Java versions with updated APIs.

Suhas Bhairav AI Resource

Pragmatic, AI-assisted migration plan to safely modernize code in weeks.

Discovery, Risk, and Scope

In this phase, map the legacy landscape, identify safe candidates, define success metrics, and establish guardrails and ownership.

Key move **Lock down data access and protect secrets before refactoring begins.**

PRACTICAL CHECKLIST

01	Catalog all legacy modules, languages, versions, dependencies, and business-critical paths.
02	Assess risks: security, data sensitivity, compliance, and external integrations.
03	Define success metrics: migration coverage, defect rate, and test reliability.
04	Select safe pilot candidates with clear ownership and observable impact.
05	Choose tooling: AI refactor assistants, static analyzers, unit-test generators, and CI integration.
06	Set guardrails: mandatory code reviews, automated tests, and a rollback plan.
07	Outline a 4-week weekly experiment plan with milestones.

Suhas Bhairav AI Resource

Pragmatic, AI-assisted migration plan to safely modernize code in weeks.

Refactor, Test, Deploy Safely

In this phase, AI-assisted refactors translate safe changes into code updates with iterative testing. Every change undergoes automated checks and human review.

Key move **Do not deploy until all critical tests pass in staging.**

PRACTICAL CHECKLIST

01	Automate safe refactors: syntax upgrades, dependency pinning, and API adjustments via AI tools.
02	Generate and run unit tests to cover migrated paths and edge cases.
03	Use shadow or staging environments to validate behavior without affecting production.
04	Enforce security checks: static analysis, dependency scans, and secure defaults.
05	Review results with owners; approve or rollback changes based on pass/fail criteria.
06	Document changes and decision rationale for future maintenance.
07	Measure progress: migration rate, defect density, and deployment frequency.

Suhas Bhairav AI Resource

Pragmatic, AI-assisted migration plan to safely modernize code in weeks.

ABOUT THE CREATOR

Suhas Bhairav

I build AI systems that combine distributed architecture, retrieval, knowledge graphs, security, and practical workflow design. My focus is AI that can be used repeatedly by real teams: understandable, reliable, and grounded in the work people already do.

EXPERTISE

- Principal systems architecture and applied AI engineering
- Production-grade AI systems, agentic workflows, RAG, and knowledge graphs
- Full-stack workflow applications across AWS, Google Cloud, Azure, on-premise, and hybrid environments
- Practical AI implementation patterns for automotive, manufacturing, finance, logistics, cybersecurity, sales, and customer operations
- Security, governance, observability, approvals, and failure-mode thinking for operational AI systems

WEBSITE

suhasbhairav.com

EMAIL

suhasbhairav@gmail.com