

ChangeFlow

A change-management workspace for modern IT teams

Deep UX Case Study

Research → Define → Product Strategy → IA → User Flows → Wireframes → High-Fidelity UI → Validation

CORE IDEA

Give teams one clear view of every change — from development and testing to approval, deployment and outcome.

Portfolio project • B2B SaaS • Desktop-first web application

01 — Executive Summary

Product: ChangeFlow is a desktop-first B2B SaaS product that helps IT teams understand, manage and track software changes from development through production.

Research: Primary research with a 5-member IT team, supported by secondary research into broader change/release-management patterns.

Central finding: The team already had a workable development-to-production process. The bigger challenge was visibility and coordination when multiple changes moved through GitHub, communication channels, staging and production at the same time.

Design response: ChangeFlow acts as a management and visibility layer around the change lifecycle instead of replacing GitHub.

Deliverable: A complete desktop SaaS concept with 10 core product screens, supporting states and a reusable design system.

Evidence note

No quantitative usability metrics are fabricated. Where validation is described, this case study documents the method and intended evaluation rather than inventing scores.

02 — Context & Problem

Software changes often pass through several people and tools before reaching production. The development workflow itself can be understandable, but the surrounding coordination becomes harder as the number of simultaneous changes grows.

In the researched team, changes moved through discussion, PRD, development, developer testing, staging, tester validation, review/approval and production. Approvals were handled through email and Microsoft Teams, while GitHub was the main place to identify and manage code changes.

The research did not suggest that the organization needed a completely new development process. Instead, it exposed a visibility problem: the information needed to understand a change was distributed across different places.

Problem statement

Teams know how to move a change to production, but managing multiple changes at once makes it difficult to know what is ready, what was tested, who approved it and what caused an issue.

03 — Research

Primary research with a 5-member IT team

Area	Observed answer
Who creates changes?	A team-level developer.
Who reviews?	Developer lead for code; manager for some application changes.
Who approves?	Developer lead.
Who implements?	Developer lead.
Who monitors?	Business owner.
Workflow	Team discussion → PRD → development → developer testing → staging → tester → review/approval → production.
Approval channels	Email and Microsoft Teams groups.
Status tracking	GitHub was the main place used currently.
Information loss	Important context can get buried in chat/communication threads.
Approval delay	Lead may be busy with other work.
Mistakes	Insufficient testing and many changes being approved/tested together.
Hard to find	What caused a production error.
Approver needs	What changed, why, affected files/components and testing context.
Risk signal	Core/main files affected was one current signal.
Priority	Severity.
Success	Production works normally.

Follow-up findings

The team clarified that quality expectations include no build errors, no merge conflicts and proper commenting. A change is considered ready from a testing perspective when all test cases tied to the PRD have passed.

Critical insight: when many PRs are approved and tested together, identifying the change responsible for a production issue can become difficult.

04 — Research Synthesis

Theme	Implication
Visibility	Users need to know where a change is without searching across tools.
Approval confidence	Approvers need concise, decision-ready context.
Testing clarity	Everyone should be able to tell whether required tests have passed.
Traceability	PR, testing, approval, deployment and outcome should stay connected.
Parallel changes	Overlapping changes should be easier to see.
Post-production learning	Failed changes should retain their history and outcome.

Key insight

IT teams don't necessarily need a new development process. They need better visibility and management around the process they already follow.

Secondary research was used as supporting validation for broader patterns such as approval bottlenecks, change collisions, testing visibility, failed changes and emergency changes; it did not override the primary research.

05 — Users & Jobs-to-be-Done

User	Need	Friction
Change Owner / Developer	Create and track a change; connect the PR; keep context together.	Repeating the same explanation across multiple places.
Approver / Tech Lead	Understand impact, risk and testing quickly.	Limited time + fragmented context.
Tester / QA	Know what to validate and record results.	Multiple changes can make ownership/status unclear.
Business Owner	Understand what changed and whether production is healthy.	Too much technical detail or insufficient visibility.

Jobs-to-be-Done

- Developer: release a change with enough context and track it to production.
- Approver: understand impact, risk and testing so a confident decision is possible.
- Tester: validate the right cases and make readiness visible.
- Business owner: understand what happened after production.

06 — Define & Product Strategy

How might we...

Help IT teams manage changes from development to production with clear visibility and less coordination overhead.

Product vision

ChangeFlow gives IT teams one clear view of every change — from development and testing to approval and production.

Positioning: GitHub remains where developers work. ChangeFlow becomes the management and visibility layer around the change.

Core product object

Change → PR → Testing → Approval → Deployment → Outcome

07 — MVP Definition

Feature	Why it belongs
Change creation	Creates the single source of truth.
Change overview	Makes current state and key context visible.
GitHub / PR connection	Keeps code context connected without recreating GitHub.
Testing tracking	Shows whether required test cases have passed.
Approval workspace	Reduces context switching for approvers.
Risk & readiness	Explains risk and what is still missing.
Change calendar	Makes overlapping changes visible.
Deployment status	Shows rollout state.
Production outcome	Closes the loop and preserves the result.
Emergency change path	Supports urgent changes while preserving traceability.

Out of scope

Not a code-hosting platform, full project-management tool, full test-management suite or complete incident-management system. Advanced AI risk prediction was also outside the first MVP.

08 — Information Architecture

Navigation	Purpose
Dashboard	Attention items, pending approvals, testing, upcoming deployments and activity.
Changes	Central workspace for all changes.
Approvals	Focused decision queue.
Calendar	Deployment windows and potential conflicts.
Insights	Management-level patterns and performance.
Settings	Workspace, environments, change types, permissions and integrations.

The Change is the single source of truth. Users do not need a separate mental model for each lifecycle stage.

09 — Primary User Flow

Stage	User action	System response
Create	Create change + essential context.	Create change record.
Connect	Link GitHub PR.	Show PR and affected code.
Validate	Run developer/tester validation.	Track cases and readiness.
Review	Approver checks context/risk/testing.	Approve or request changes.
Schedule	Select deployment window.	Add to calendar.
Deploy	Run deployment.	Show live state/logs.
Validate	Confirm production health.	Record post-deployment checks.
Close	Record outcome.	Preserve complete history.

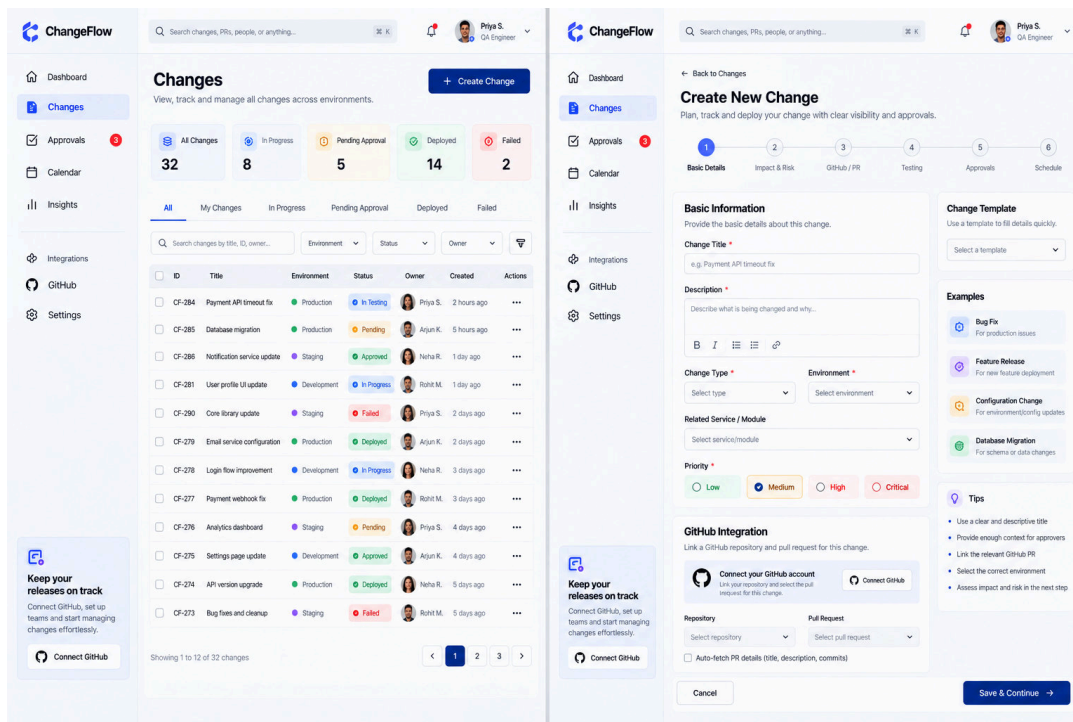
Interaction principle

A change is a living record, not a rigid checkout flow. Users can return later and continue from its current state.

10 — Low-Fidelity Wireframes

The wireframe phase focused on information hierarchy, task completion and lifecycle clarity before visual styling. The complete product was mapped rather than designing only portfolio showcase screens.

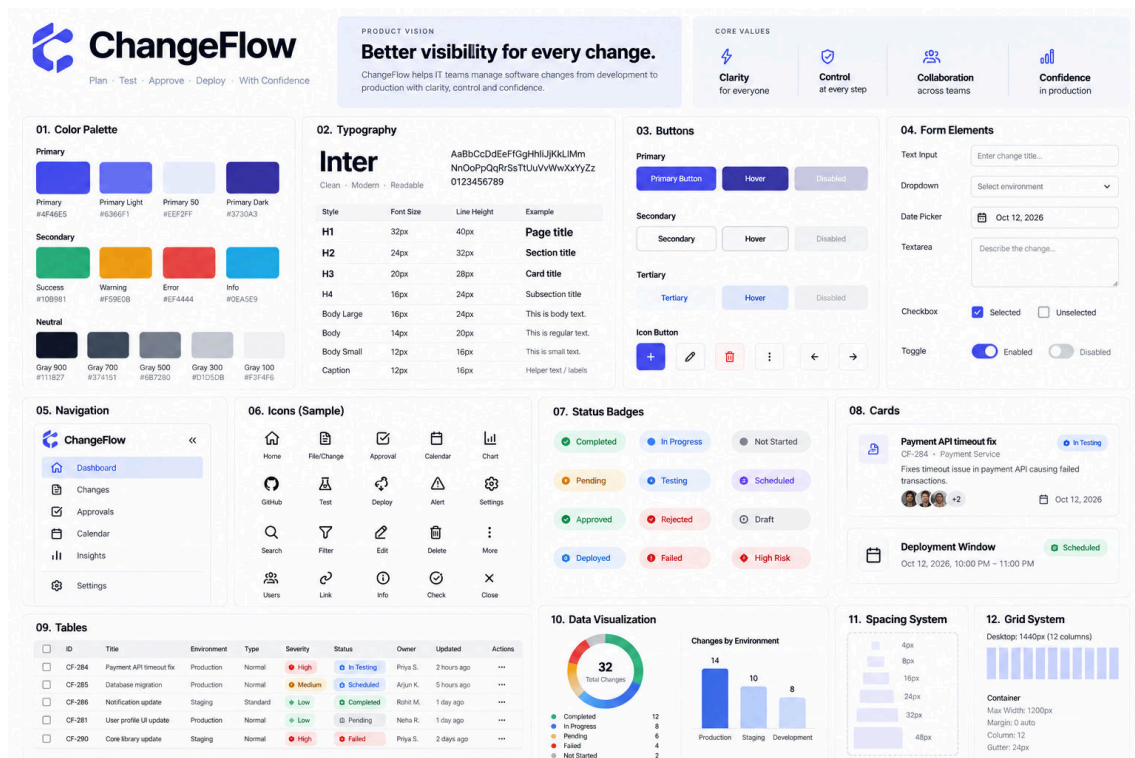
Core coverage: Dashboard • Changes • Create Change • Change Details • Impact & Risk • GitHub / PR • Testing • Approvals • Calendar • Deployment • Production Outcome • Activity / History



Example wireframe structure: Changes list and Create Change.

11 — Visual Design System

The visual system was designed for a desktop-first B2B SaaS product: clear hierarchy, restrained color, high information density without visual noise, and strong status communication.

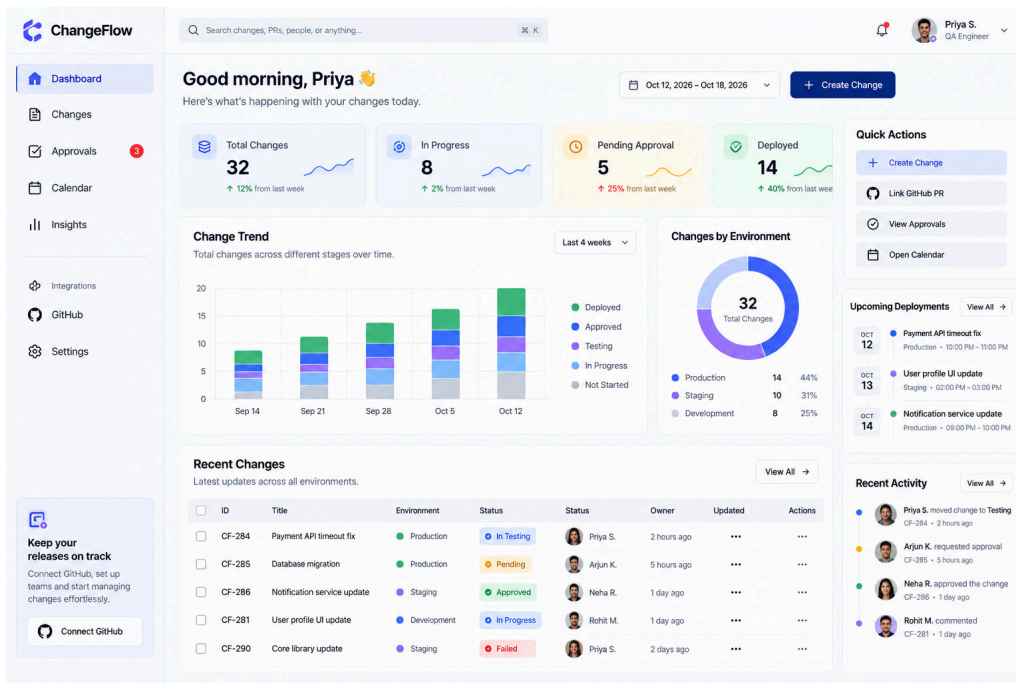


ChangeFlow design-system board.

12 — High-Fidelity Screens

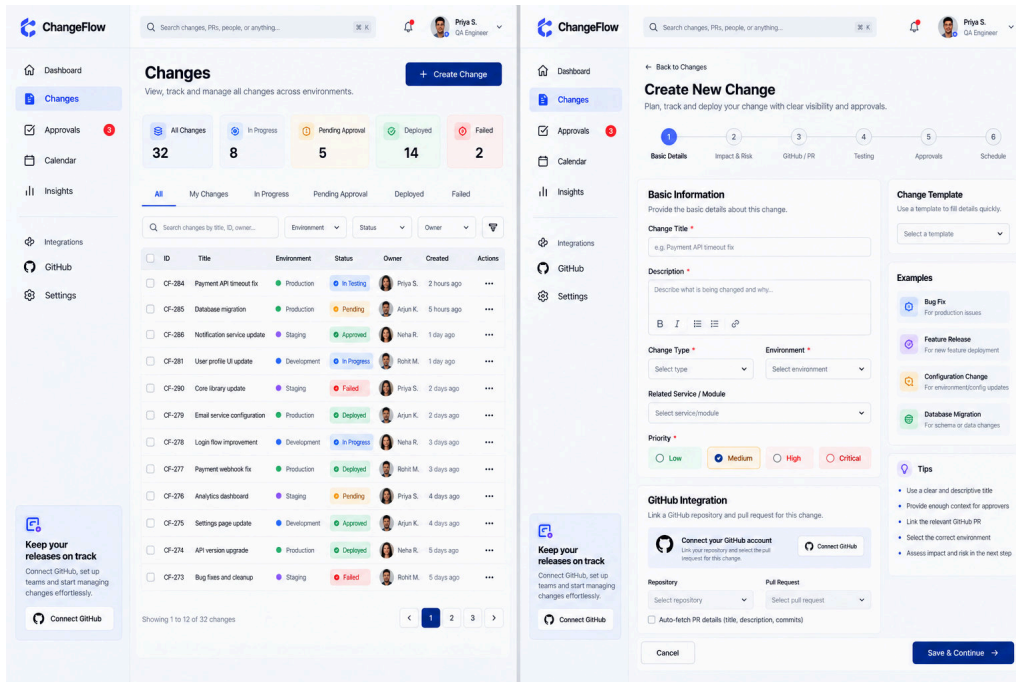
The high-fidelity phase translated the wireframe structure into a cohesive web application with a light workspace, rounded cards, subtle shadows, blue/indigo actions and semantic status colors.

Dashboard



Operational overview.

Changes + Create Change



Change list and creation workflow.

Change Details + Approvals

The image displays two side-by-side screenshots of the ChangeFlow application interface, illustrating the 'Change Details' and 'Approvals' sections.

Left Screenshot: Change Details

- Header:** ChangeFlow logo, search bar, user profile (Priya S., QA Engineer).
- Navigation:** Dashboard, Changes, Approvals (3), Calendar, Insights, Integrations, GitHub, Settings.
- Change Details:**
 - CF-284 Payment API timeout fix** (In Testing, Medium Risk, Normal Change).
 - Description:** Payment API is returning timeout errors during high traffic which causes failed transactions. This change optimizes the API request handling and increases the timeout limit from 5s to 30s.
 - Change Progress:** Created (Oct 10, 2026, 11:24 AM), Impact Analysis (Oct 10, 2026, 01:12 PM), GitHub PR (Oct 10, 2026, 03:45 PM), Testing (Oct 11, 2026, 10:20 AM - In Progress), Approvals (Pending), Deployment (Not Started), Closed (Not Started).
 - Change Details:** Change Type: Bug Fix, Environment: Production, Related Service: Payment Service, Requested By: Priya S. (QA Engineer), Owner: Arjun K. (Backend Developer), Priority: Medium, Planned Start: Oct 12, 2026, 10:00 PM, Planned End: Oct 12, 2026, 11:00 PM, Related Incident: INC-1123.
 - Attachments:** timeout-error-logs.txt (12 KB), proposed-fix-details.pdf (250 KB), api-performance-chart.png (450 KB).
 - People Involved:** Priya S. (Requestor, QA Engineer), Arjun K. (Owner, Backend Developer), Neha R. (Approver, Tech Lead), Rohit M. (Reviewer, DevOps Engineer).
 - Tags:** payment, api, timeout.
- Footer:** Keep your releases on track. Connect GitHub, set up teams and start managing changes effortlessly. Connect GitHub button.

Right Screenshot: Approvals

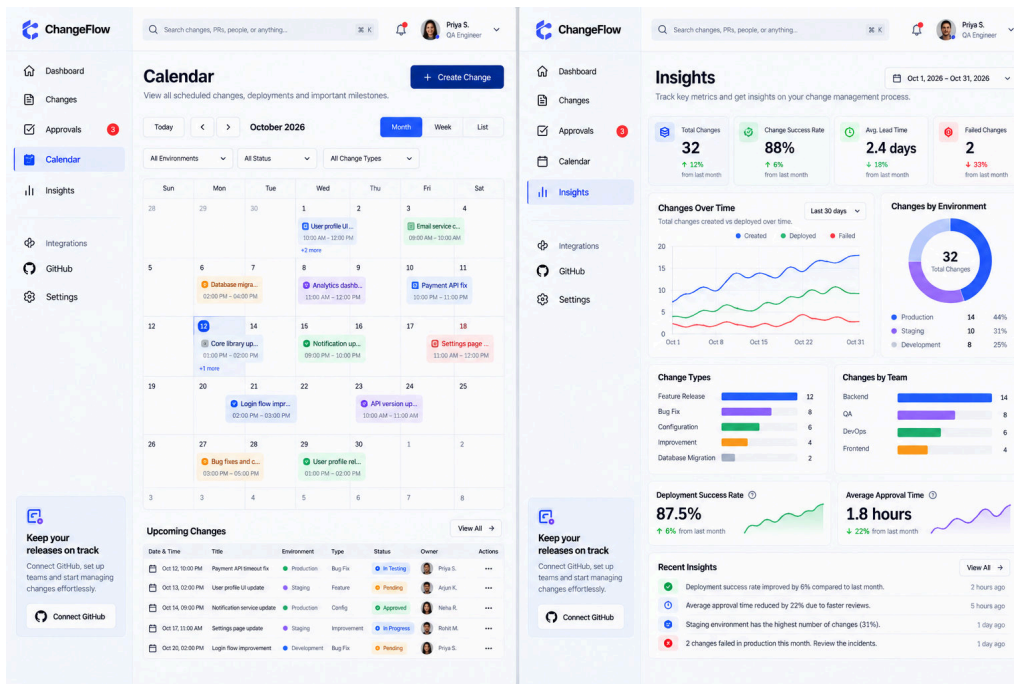
- Header:** ChangeFlow logo, search bar, user profile (Priya S., QA Engineer).
- Navigation:** Dashboard, Changes, Approvals (5), Calendar, Insights, Integrations, GitHub, Settings.
- Approvals:**
 - CF-284 Payment API timeout fix** (Medium Risk, Normal Change). Requested 2 hours ago. Approve/Reject buttons.
 - CF-285 Database migration** (High Risk, Standard Change). Requested 8 hours ago. Approve/Reject buttons.
 - CF-291 Analytics dashboard update** (Low Risk, Normal Change). Requested 1 day ago. Approve/Reject buttons.
 - CF-293 Email service configuration** (Medium Risk, Standard Change). Requested 1 day ago. Approve/Reject buttons.
 - CF-295 User profile feature release** (Low Risk, Normal Change). Requested 2 days ago. Approve/Reject buttons.
- Footer:** Keep your releases on track. Connect GitHub, set up teams and start managing changes effortlessly. Connect GitHub button.

Approval Workflow Diagram:

```
graph LR; 1[1. Request Submitted By Requestor] --> 2[2. Technical Review Tech Lead]; 2 --> 3[3. Security Review Security Team]; 3 --> 4[4. Final Approval Change Manager];
```

Central change record and approval queue.

Calendar + Insights



Scheduling and management visibility.

Testing + Deployment

The image displays two side-by-side screenshots of the ChangeFlow web application interface, focusing on the 'Testing' and 'Deploy Change' sections.

Left Screenshot (Testing):

- Header:** ChangeFlow logo, search bar, user profile (Priya S., QA Engineer).
- Left Sidebar:** Dashboard, Changes, Approvals (3), Calendar, Testing (selected), Deployments, Insights, Integrations, GitHub, Settings.
- Main Content:**
 - Testing Section:** Run tests, track results and ensure quality before deployment. View Testing Guidelines.
 - Change Card:** CF-284 Payment API timeout fix. Status: In Testing. Labels: Production, Medium Risk, Bug Fix. Description: Fixes timeout issue in payment API causing failed transactions. Updated 2 hours ago.
 - Summary:** Total Test Cases: 24 (100% created). Passed: 18 (75% passed). Failed: 4 (17% failed). Blocked: 2 (8% blocked).
 - Test Cases Table (24):**

ID	Title	Type	Priority	Status	Executed By	Actions
TC-001	Validate API timeout error	Functional	High	Passed	Priya S.	...
TC-002	Retry mechanism check	Functional	High	Passed	Ajay K.	...
TC-003	Error message validation	Functional	Medium	Failed	Neha R.	...
TC-004	Payment success flow	Regression	High	Passed	Rohit M.	...
TC-005	Load testing (1000 req)	Performance	High	Blocked	Priya S.	...
TC-006	Database rollback	Functional	Medium	Passed	Ajay K.	...
TC-007	Invalid payload handling	Negative	Medium	Failed	Neha R.	...
TC-008	API response time	Performance	Low	Passed	Rohit M.	...
 - Latest Test Run:** Test Run #12. Executed on Oct 12, 2026, 11:30 AM by Priya S. Status: Completed. Summary: 18 Passed, 4 Failed, 2 Blocked.
 - Environments:**
 - Development: v2.4.0 (Active)
 - Staging: v2.4.0 (Active)
 - Production: v2.3.1 (Inactive)
- Bottom Bar:** Keep your releases on track. Connect GitHub.

Right Screenshot (Deploy Change):

- Header:** ChangeFlow logo, search bar, user profile (Priya S., QA Engineer).
- Left Sidebar:** Dashboard, Changes, Approvals (3), Calendar, Testing, Deployments (selected), Insights, Integrations, GitHub, Settings.
- Main Content:**
 - Deploy Change Section:** Deploy approved changes to production with complete tracking and rollback support. View Change.
 - Change Card:** CF-284 Payment API timeout fix. Status: Approved. Labels: Production, Medium Risk, Bug Fix. Version: v2.4.0. Target Environment: Production. Description: Fixes timeout issue in payment API causing failed transactions. Approved 5 hours ago.
 - Deployment Progress:** Prepare (1) - Deploy (2) - Verify (3) - Complete (4). Current status: Deploying to servers (In Progress, 68%).
 - Deployment Configuration:**
 - Environment: Production
 - Version: v2.4.0
 - Deployment Type: Rolling Update
 - Schedule: Oct 12, 2026, 10:00 PM
 - Rollback Plan: Automatic (v2.3.1)
 - Notify: Releases, Backend-team
 - Deployment Logs:** Live Logs. Log content: [22:40:18] Starting deployment for v2.4.0, [22:40:19] Running pre-deployment checks, [22:40:24] All checks passed, [22:40:25] Pulling latest image, [22:40:26] Deploying to production servers, [22:40:27] Updating configurations, [22:40:42] Health checks in progress, [22:40:43] 2 of 3 servers returned 200, [22:40:48] Continuing deployment...
 - Rollback Options:** Rollback to v2.3.1. Revert to the previous stable version if any issues occur during deployment.
 - Post Deployment Checks:**
 - API Health Check: Passed
 - Database Connection: Passed
 - Payment Flow Test: In Progress
 - Error Log Scan: Pending
 - Response Time: In Progress
 - Queue Processing: Pending
 - Cache Status: Pending
 - External Services: Pending
- Bottom Bar:** Keep your releases on track. Connect GitHub.

Quality validation and production rollout.

Settings

ChangeFlow

Search changes, PRs, people, or anything...

K

Dashboard

Changes

Approvals 3

Calendar

Testing

Deployment

Insights

Integrations

GitHub

Settings

Keep your releases on track

Connect GitHub, set up teams and start managing changes effortlessly.

Connect GitHub

Settings

Manage your workspace settings, team members, environments and integrations.

General

Environments

Change Types

Integrations

Notifications

Access Control

Audit Logs

Workspace Details

Update your organization details and preferences.

Save Changes

ChangeFlow

Change Logo

Organization Name *

ChangeFlow

Organization Description

Change management platform for modern development teams.

Time Zone

(GMT +05:30) India Standard Time

Date Format

Oct 12, 2026

Environments

Manage your deployment environments.

+ Add Environment

Name	Description	Color	Actions
Production	Live production environment	Production	
Staging	Pre-production testing environment	Staging	
Development	Development and feature testing	Development	
QA	Quality assurance environment	QA	

Change Types

Customize change types for your organization.

+ Add Change Type

Name	Description	Color	Risk Level	Actions
Feature Release	New feature or enhancement		Low	
Bug Fix	Fix for a bug or issue		Medium	
Configuration Change	Infrastructure or configuration updates		Medium	
Database Migration	Database schema or data changes		High	

Default Settings

Configure default preferences for your workspace.

Default Environment

Production

Default Change Type

Feature Release

Default Priority

Medium

Auto-assign reviewers

Automatically assign reviewers based on code owners

Require test cases

Make test cases mandatory for all changes

Enable automatic notifications

Send notifications for important updates

Working Hours

Mon - Fri (9:00 AM - 6:00 PM)

Language

English

Team Members

Manage people, roles and permissions.

+ Invite Member

Priya Sharma

priya@company.com

Admin

...

Arjun Kumar

arjun@company.com

Developer

...

Neha Reddy

neha@company.com

QA Engineer

...

Rohit Mehta

rohit@company.com

DevOps

...

Sneha Patil

sneha@company.com

Viewer

...

View All Members

Integrations

Connect your development tools and services.

GitHub

Connect your repository for PR tracking and automation.

Connected

>

Jira

Sync issues and track change requests.

Connect

>

Slack

Get notifications about changes and approvals.

Connected

>

Microsoft Teams

Get notifications in your Teams channels.

Connect

>

Jenkins

Integrate with your CI/CD pipeline for automated deployments.

Connect

>

SonarQube

Get code quality and security analysis.

Connect

>

View All Integrations

Workspace administration and integrations.

ChangeFlow — Deep UX Case Study

18

13 — Final 10-Screen Product Scope

#	Screen	Primary job
01	Dashboard	See what needs attention.
02	Changes	Find, filter and manage changes.
03	Create Change	Create a structured change record.
04	Change Details	Understand one change end-to-end.
05	Approvals	Review and decide.
06	Testing	Track test cases and readiness.
07	Calendar	See deployment windows and conflicts.
08	Insights	Understand operational patterns.
09	Deployment	Manage and monitor rollout.
10	Production Outcome	Validate, record outcome and close.

Supporting states include pending, in-progress, failed, approved, requested changes, scheduled, deployed, rollback and emergency-change states.

14 — Usability Validation

Validation goal

Can an IT user understand a change, complete the required action and move it through development → testing → approval → production without unnecessary confusion?

Recommended participant mix

Two developers, one QA/tester, one tech lead/manager and one person familiar with software projects. The original 5-member IT team is especially valuable.

Task scenarios

- Find the current status of the Payment API timeout fix.
- Find what is changing, why, affected service and current risk.
- Verify whether testing is complete and whether anything failed.
- Approve the change after reviewing the required context.
- Use Activity/History to understand what happened before a production issue.

What to observe

- Lifecycle/status clarity.
- Approval confidence and decision context.
- Testing comprehension.
- Ability to trace PR → testing → approval → deployment.
- Information overload on Change Details.

Evidence discipline

This document does not invent task-completion percentages, SUS scores or time-on-task results. Those should only be added after actual sessions are recorded.

15 — Accessibility & Interaction Principles

Principle	Decision
Hierarchy	Strong titles and progressive disclosure prevent dense enterprise screens from becoming visually flat.
Status	Color is paired with labels/icons.
Interaction	Controls use comfortable click targets.
Contrast	High-contrast neutral text with restrained semantic colors.
Consistency	Reusable buttons, badges, inputs, tables and navigation.
Traceability	Important actions leave visible history.

Responsive strategy

Desktop is primary because users work with tables, approvals, testing evidence and deployment schedules. Smaller breakpoints should collapse secondary navigation and convert wide tables into focused views.

16 — Reflection & Portfolio Takeaway

What makes ChangeFlow a deep UX project?

- The product starts from a real workflow rather than an invented interface.
- Primary research directly influenced the problem definition.
- Secondary research validated broader patterns without overriding team evidence.
- The solution complements GitHub instead of replacing it.
- The Change object creates a consistent mental model.
- The full app architecture is designed, not only five showcase screens.

Final portfolio statement

I designed ChangeFlow, a desktop-first change-management workspace, after researching how an IT team moves software changes from development to production. The research revealed that the biggest challenge was not the process itself, but fragmented visibility when multiple changes moved through GitHub, testing, communication and production together. I translated that insight into a change-centered product architecture, end-to-end workflow and high-fidelity SaaS experience that connects PRs, testing, approvals, deployment and outcome in one place.

END OF CASE STUDY