PLATFORM DOCUMENTATION

CorePlexML

Release 0.1.0

Rodrigo Henriquez M.

Generated on Mar 09, 2026

CONTENTS

Audience: platform users operating CorePlexML via web UI.

CHAPTER

ONE

1. ACCOUNT ONBOARDING

1.1 Goal

Create a valid user account, complete first login, and confirm workspace access.

1.2 Who should read this

1. New users in Free, Pro, or Team plans.
2. Team members invited by an owner/admin.

1.3 Preconditions

1. Access to CorePlexML web URL.
2. Valid email inbox.
3. Browser with cookies and JavaScript enabled.

1.4 Step-by-step

1. Open `/register`.
2. Complete required fields: - Full name. - Work email. - Password (platform policy). - Terms acceptance.
3. Submit registration.
4. If email verification is enabled, open the verification email and confirm.
5. Open `/login` and authenticate.
6. Complete first-session profile fields if prompted: - Display name. - Timezone. - Default workspace/project (if available).
7. Confirm dashboard loads without auth errors.

1.5 Functional validation checklist

1. User session is active after login.
2. User menu shows correct account email.
3. Dashboard widgets load with HTTP 200 responses.
4. Logout ends session and redirects to login page.

5. Re-login restores valid session and dashboard access.

1.6 Expected result

1. Authenticated user can navigate all modules allowed by role.
2. Session state is stable across page refresh.

1.7 Common errors and recovery

1. `Email not verified`: - Resend verification email. - Confirm verification link opens same environment domain.
2. `Invalid credentials`: - Use `/auth/forgot-password`. - Retry with updated password.
3. Immediate logout after login: - Clear browser cookies for domain. - Retry and verify system clock/timezone on client.

1.8 Screenshot

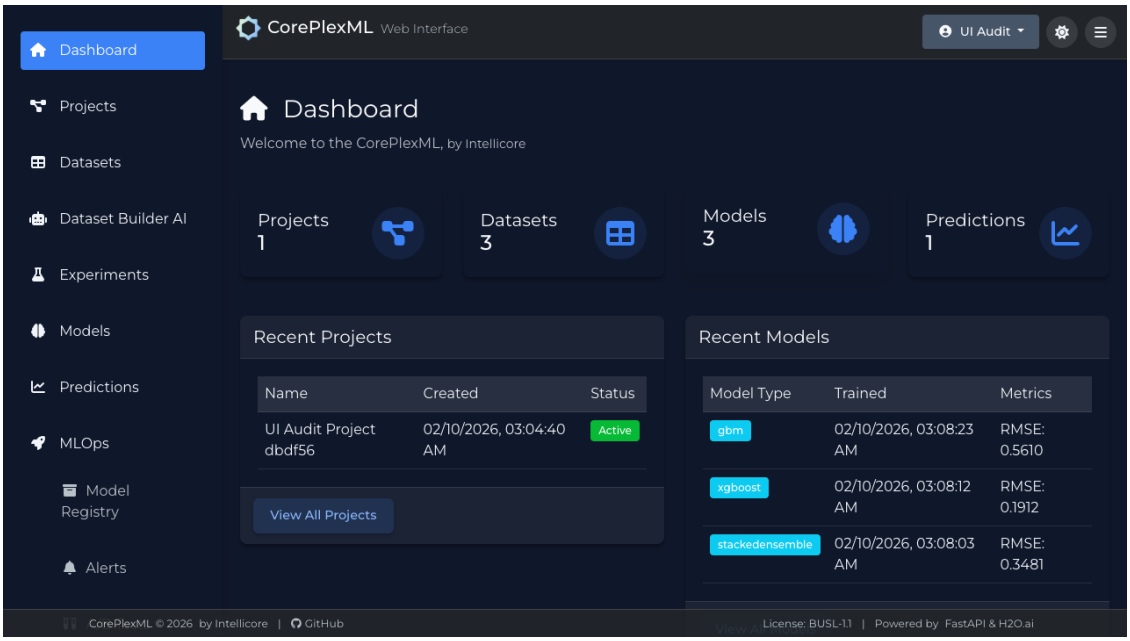


Fig. 1: Dashboard visible after onboarding and first login.

CHAPTER

TWO

2. NAVIGATION AND WORKSPACE

2.1 Goal

Understand global navigation, active project context, and safe movement between modules.

2.2 Workspace semantics

1. Global sidebar controls module-level navigation.
2. Top bar controls search, profile, and quick actions.
3. Active project context determines where data and runs are created.

2.3 Step-by-step

1. Open dashboard.
2. Identify key navigation areas: - Left sidebar: Datasets, Builder, Experiments, Models, MLOps, Privacy, Synth-Gen. - Top bar: search, notifications, profile/team.
3. Change module from sidebar and confirm page title updates.
4. Switch active project (if selector is available).
5. Return to previous module and confirm context remains stable.

2.4 Functional validation checklist

1. Each menu item routes to the correct module URL.
2. Browser back/forward preserves expected state.
3. Active project badge/selector remains consistent across modules.
4. Forbidden modules are hidden or blocked for non-authorized roles.
5. Table filters and search survive refresh when expected by product behavior.

2.5 Expected result

1. User can move across modules without losing project context.
2. Navigation reflects role permissions and plan limits.

2.6 Common errors and recovery

1. Data appears from wrong project: - Re-check active project selector. - Hard refresh and re-enter module.
2. Menu option missing: - Verify user role and subscription plan.
3. Stale UI state: - Clear local storage for platform domain and re-login.

2.7 Screenshot

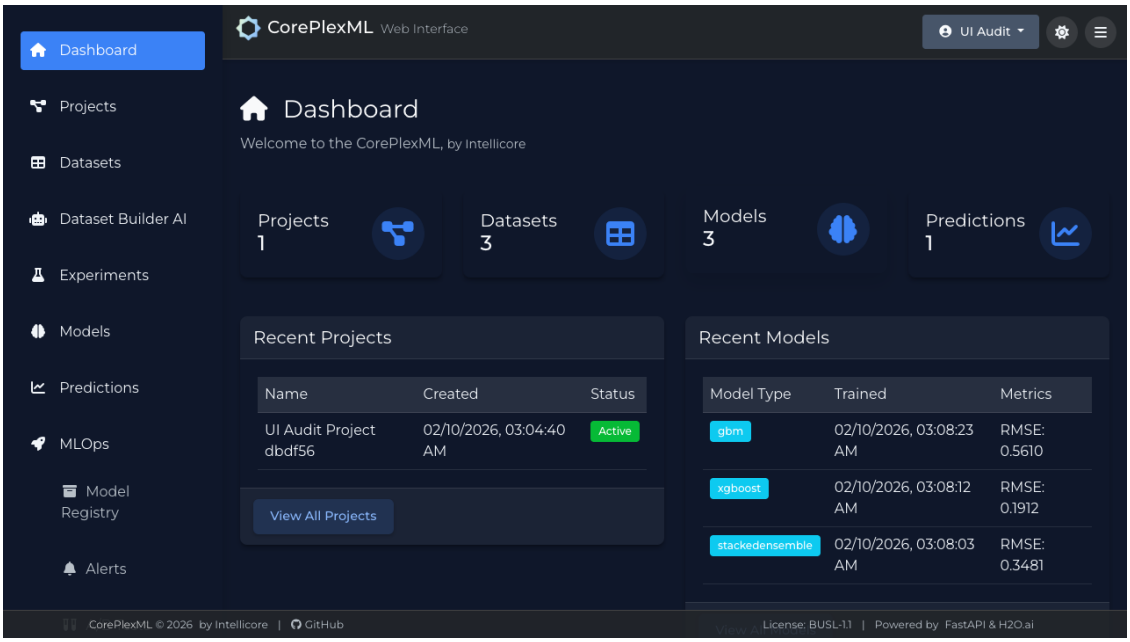


Fig. 1: Main workspace navigation with module and profile access points.

CHAPTER

THREE

3. PROJECTS AND DATASETS

3.1 Goal

Create projects, upload datasets, and validate data integrity before modeling.

3.2 Preconditions

1. User has project create/upload permissions.
2. Data file is available in CSV or Parquet format.

3.3 Create a project

1. Open `Projects`.
2. Click `Create Project`.
3. Set: - Project name. - Optional description. - Visibility/team scope if enabled.
4. Save and confirm project appears in list.

3.4 Upload a dataset

1. Open `Datasets` inside the target project.
2. Click `Upload Dataset`.
3. Select file and optional metadata fields.
4. Wait for ingestion completion status.
5. Open dataset detail view.

3.5 Validate dataset quality

1. Review schema inference: - Column names. - Data types. - Null counts.
2. Review row count and duplicate indicators.
3. Review first rows preview for parsing issues.
4. Confirm delimiters/date formats were interpreted correctly.

3.6 Functional validation checklist

1. Dataset artifact is created and visible in table.
2. Reported row count matches source file expectation.
3. Critical columns preserve expected types.
4. Preview values are not shifted/truncated unexpectedly.
5. Re-opening dataset detail returns same metadata (idempotent view).

3.7 Expected result

1. Project and dataset are ready for downstream workflows.
2. Dataset version is selectable in Builder/Experiments.

3.8 Common errors and recovery

1. Upload stuck in processing: - Retry upload with smaller sample. - Check server logs for parser error.
2. Wrong type detection: - Re-upload with cleaned headers/date format.
3. Row mismatch vs source: - Validate delimiter/quote settings and malformed rows.

3.9 Screenshots

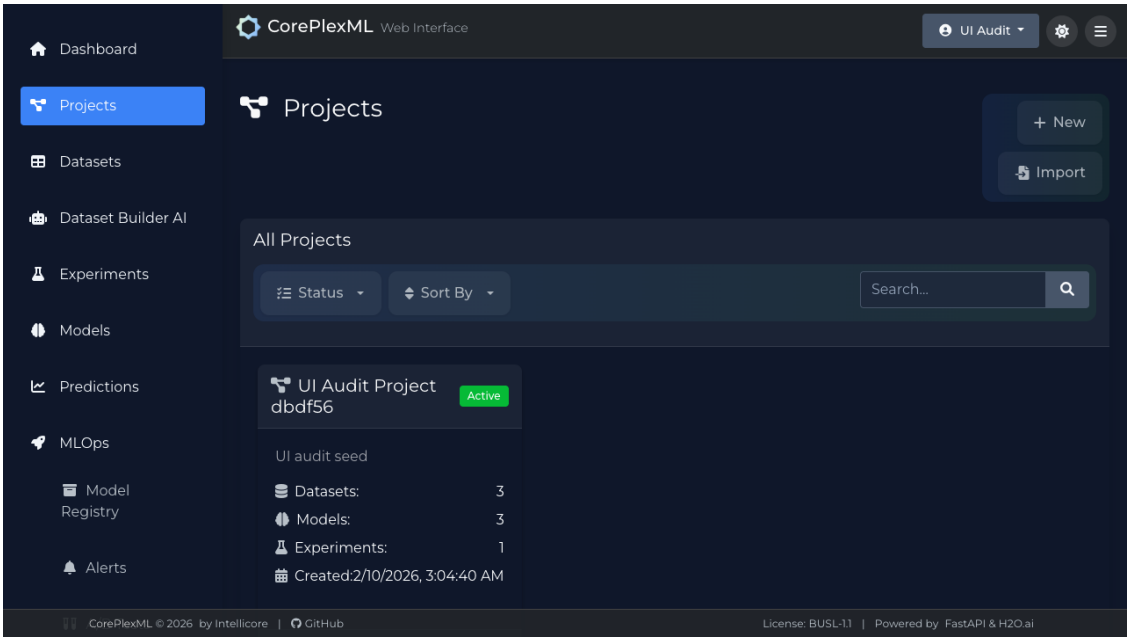


Fig. 1: Projects module with creation and selection flow.

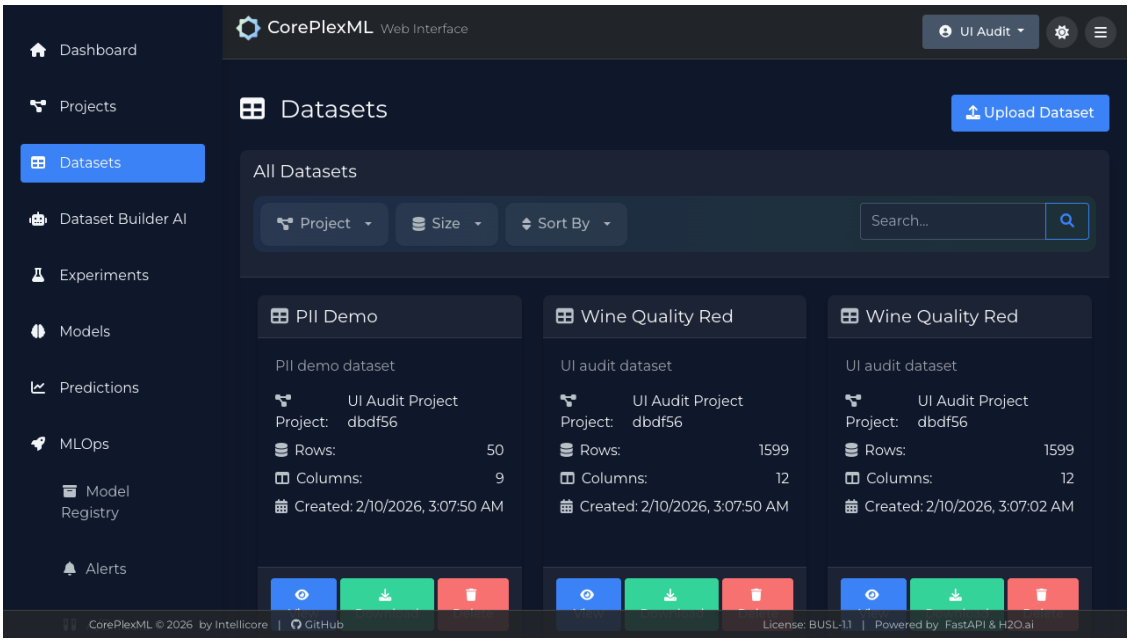


Fig. 2: Dataset registry with upload status and metadata summary.

CHAPTER

FOUR

4. DATASET BUILDER AI

4.1 Goal

Produce a transformed dataset through a guided conversational workflow with full traceability.

4.2 What this module must do

1. Ask relevant questions for the current data step.
2. Generate and execute transformations.
3. Show preview and row/column impact.
4. Allow confirm, retry, or correction paths.

4.3 Run a realistic conversation

1. Open `Dataset Builder AI` and select a dataset version.
2. Read the initial analysis summary.
3. Answer first questions with realistic intent: - Ask clarifications. - Change decision mid-flow. - Reject an outcome and request retry.
4. Continue until script executes and output preview is displayed.
5. Confirm only if output matches objective.

4.4 If output is not what you expect

1. Ask for clarification before confirming the step.
2. Adjust your previous choices and run retry.
3. If row count drops unexpectedly, retry with safer cleaning parameters.
4. If output reaches zero rows, use retry and review transformation options.

4.5 Functional validation checklist

1. Builder generates output artifact and new dataset version.
2. Output row count is non-negative and explicitly reported.
3. Preview table reflects transformation choices.

4. Conversation state persists across refresh/re-entry.
5. Retry path restores previous valid dataset when requested.

4.6 Expected result

1. User can iterate from analysis to final artifact without hidden state loss.
2. Final dataset is usable by Experiments module.

4.7 Common errors and recovery

1. `Zero-row output` warning: - Use retry path and adjust cleaning strategy.
2. Unexpected transformation result: - Request rollback/retry in conversation.
3. Session interruption: - Re-open Builder session and continue from saved state.

4.8 Screenshot

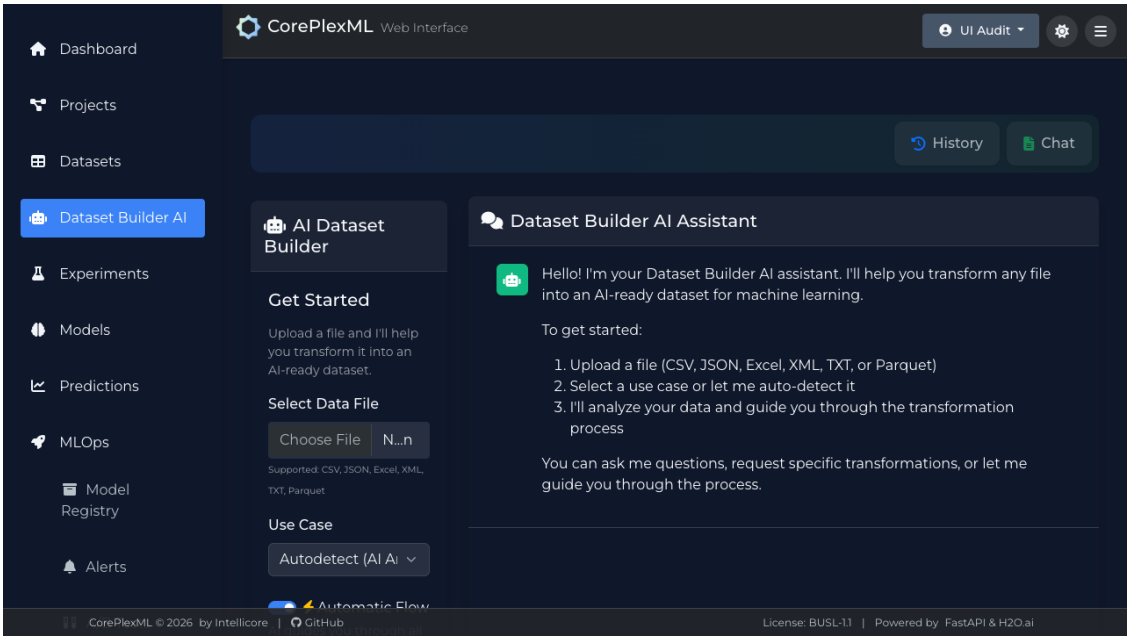


Fig. 1: Conversational flow with transformation output preview.

CHAPTER

FIVE

5. EXPERIMENTS, MODELS, PREDICTIONS

5.1 Goal

Train models, compare results, register best candidate, and run prediction workflows.

5.2 Create an experiment

1. Open `Experiments`.
2. Click `New Experiment`.
3. Configure: - Dataset/version. - Target column. - Problem type (classification/regression/time series if available). - Validation strategy and objective metric.
4. Start run and monitor status.

5.3 Review results and model registry

1. Open leaderboard when run completes.
2. Confirm primary metric and rank ordering.
3. Open best model detail and inspect: - Metrics. - Feature importance/explainability (if enabled). - Artifact/version metadata.
4. Register or pin model according to workflow.

5.4 Run predictions

1. Open `Predictions`.
2. Run single-record prediction from UI form.
3. Run batch prediction using file upload if available.
4. Validate output fields: - Predicted value/class. - Confidence/probability (if applicable). - Request timestamp/run reference.

5.5 Functional validation checklist

1. Experiment transitions to terminal state without silent failure.
2. Metrics shown in model detail match leaderboard values.

3. Prediction output schema is stable across repeated calls.
4. Batch result row count matches input record count.
5. Prediction errors return actionable messages.

5.6 Expected result

1. At least one model is ready for deployment.
2. Prediction workflow returns consistent, traceable outputs.

5.7 Common errors and recovery

1. Experiment fails early: - Validate target column and missing value handling.
2. Metric looks inconsistent: - Confirm same split/seed settings.
3. Prediction input rejected: - Align field names/types with model input schema.

5.8 Screenshots

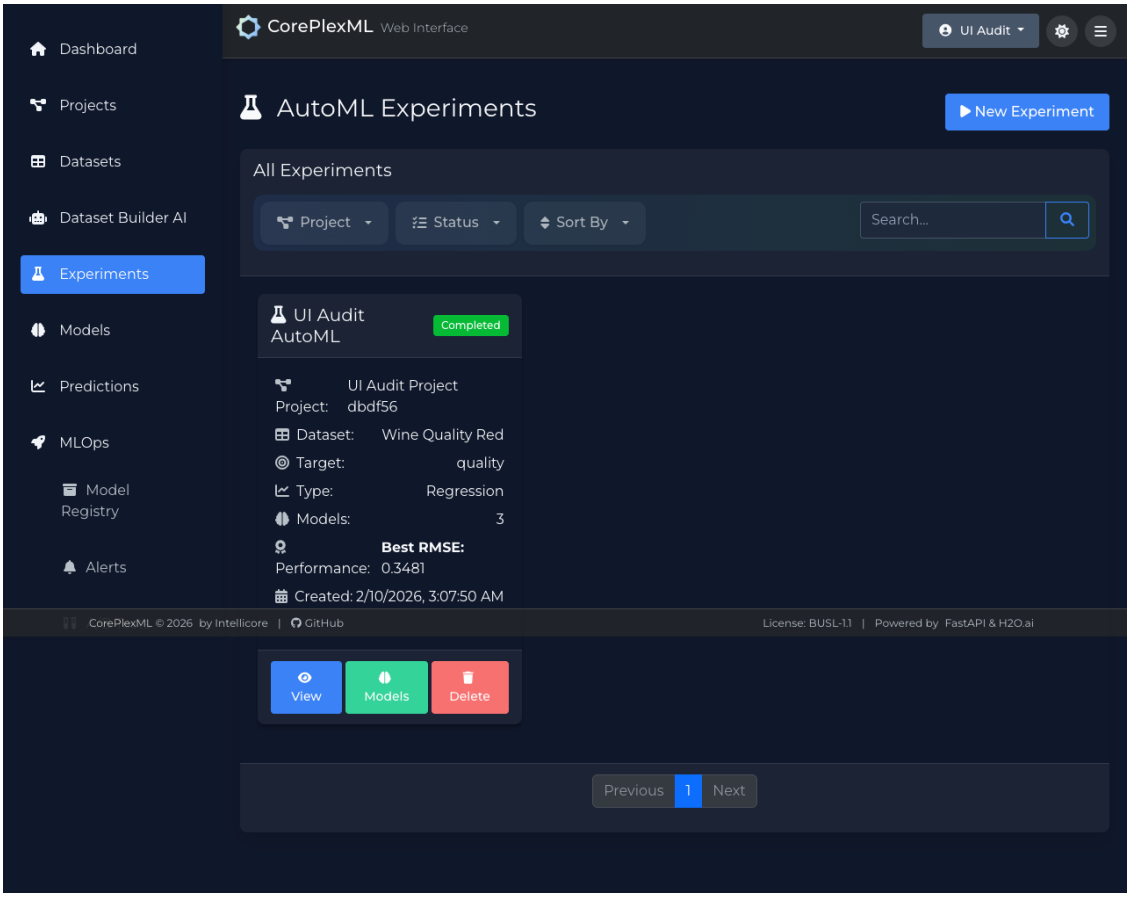


Fig. 1: Experiment execution and status monitoring.

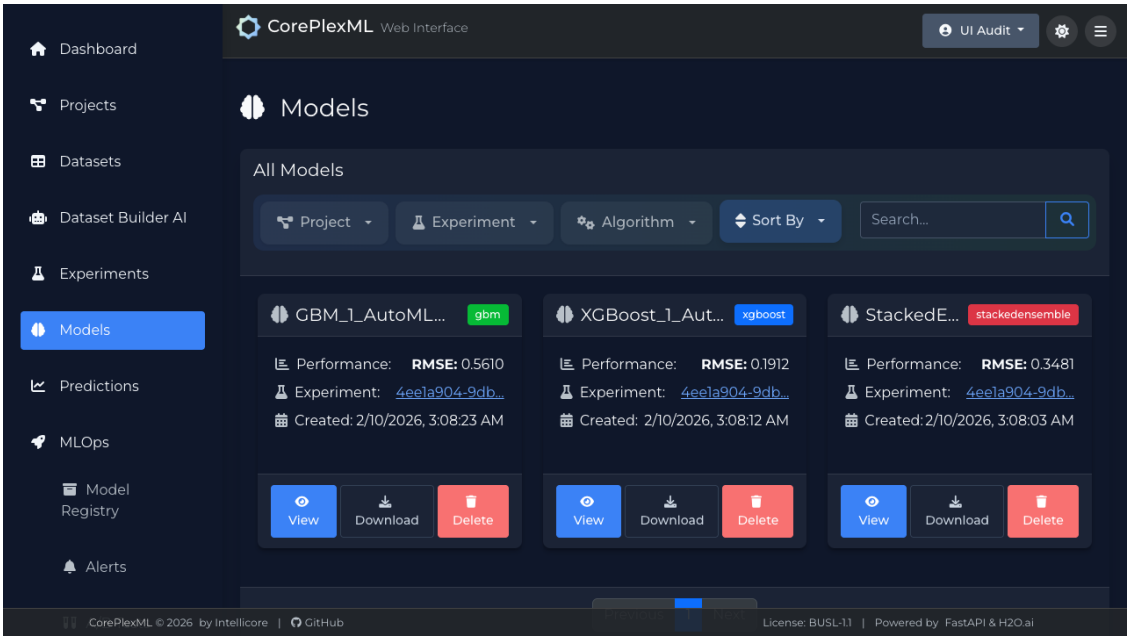


Fig. 2: Model registry with metric and artifact metadata.

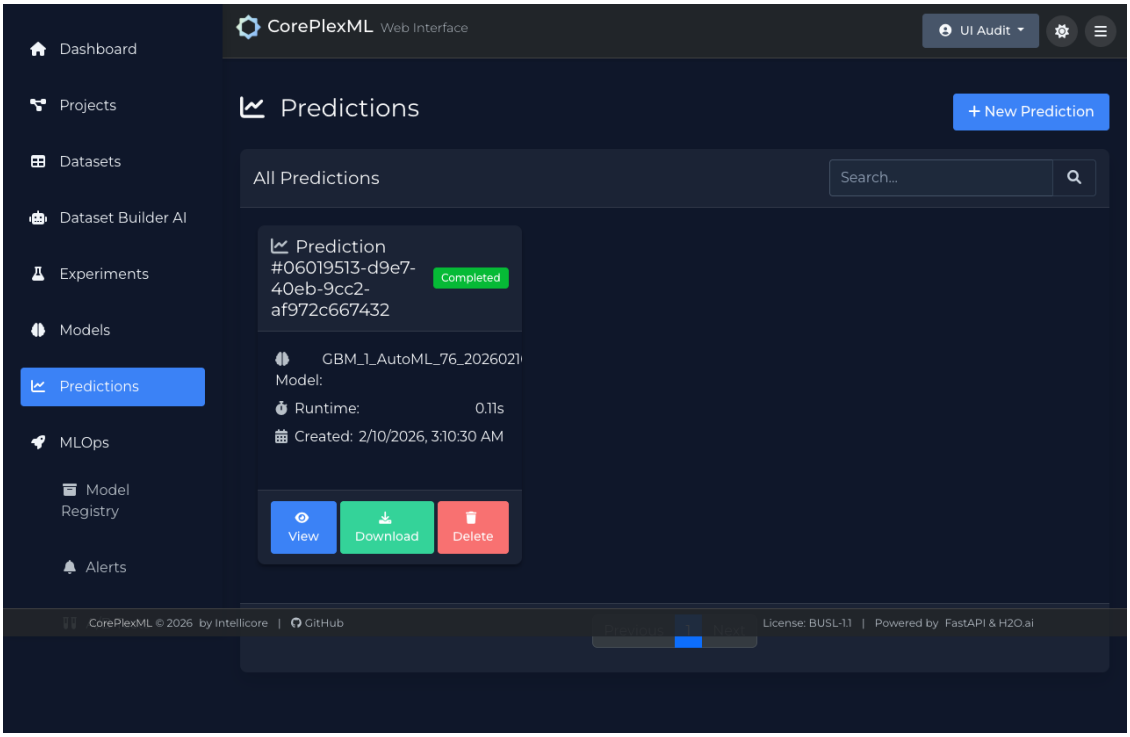


Fig. 3: Prediction UI for single and batch inference.

CHAPTER

SIX

6. MLOPS AND A/B TESTING

6.1 Goal

Deploy models safely, monitor runtime behavior, and compare variants with statistical discipline.

6.2 Create a deployment

1. Open `MLOps > Deployments`.
2. Select a model version from registry.
3. Configure deployment settings: - Environment. - Replica/compute profile. - Rollback strategy.
4. Deploy and wait until status is `active`.

6.3 Monitor deployment

1. Validate health indicators: - Uptime/health state. - Error rate. - Latency percentile.
2. Generate test inference call and confirm response.
3. Review recent logs for runtime exceptions.

6.4 Run A/B test

1. Open `MLOps > A/B Tests`.
2. Create a test with: - Baseline model (A). - Candidate model (B). - Traffic split. - Primary success metric. - Minimum sample size/duration.
3. Start test and monitor allocation.
4. Evaluate winner decision when threshold is reached.

6.5 Functional validation checklist

1. Active deployment serves predictions without downtime.
2. Health and metrics update in near real-time.
3. A/B traffic split is respected by observed request counts.
4. Reported winner is supported by configured metric.
5. Rollback can be executed if candidate degrades performance.

6.6 Expected result

1. Production path is stable and observable.
2. Model promotion decisions are data-driven.

6.7 Common errors and recovery

1. Deployment stuck in `pending`: - Check environment capacity and model artifact availability.
2. High error rate after release: - Trigger rollback to previous stable model.
3. Inconclusive A/B test: - Increase duration/sample size before decision.

6.8 Screenshots

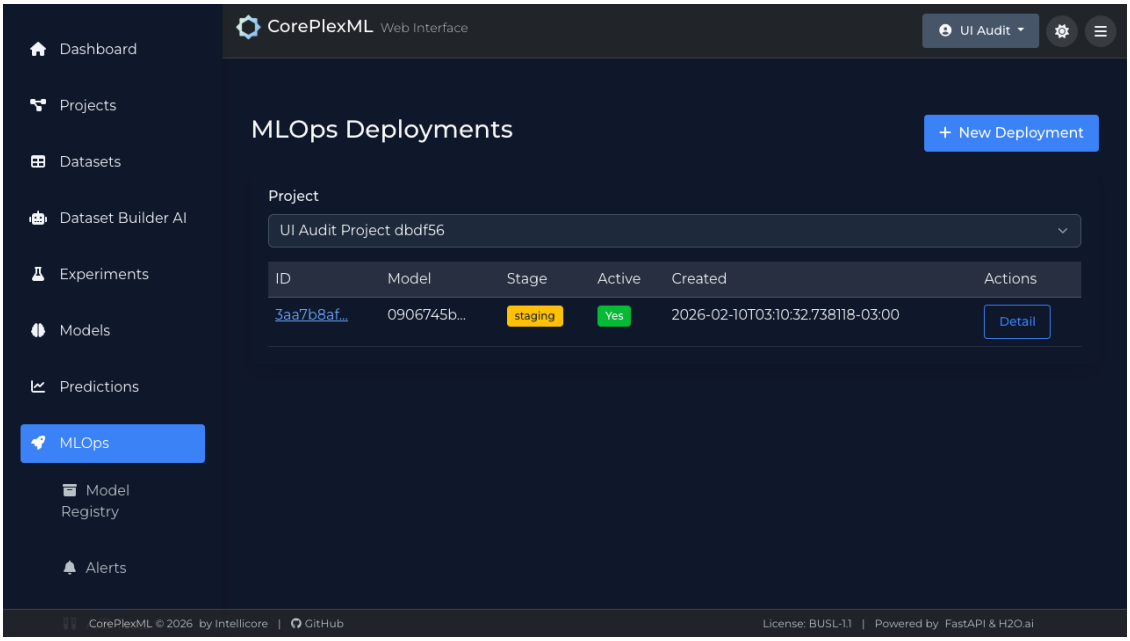


Fig. 1: MLOps deployment list with runtime status.

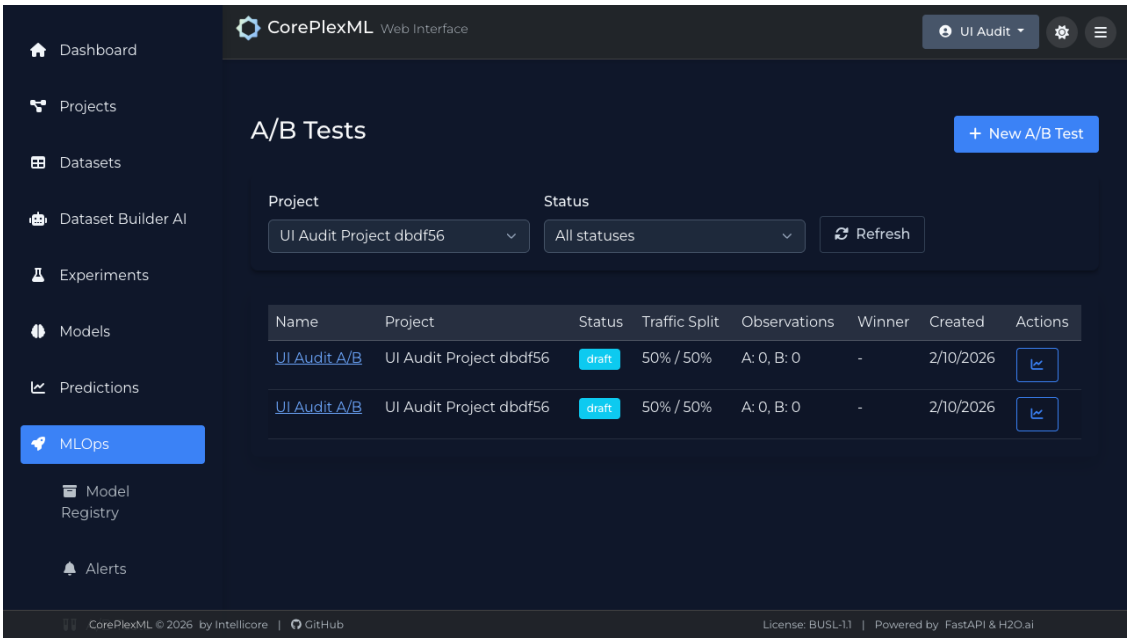


Fig. 2: A/B testing configuration and live comparison view.

CHAPTER

SEVEN

7. PRIVACY AND SYNTHGEN

7.1 Goal

Protect sensitive data with policy-driven transformations and create synthetic datasets for safe experimentation.

7.2 Privacy workflow

1. Open `Privacy`.
2. Create or select a privacy policy.
3. Attach policy to dataset/session.
4. Run detection and review discovered PII entities.
5. Configure transformation action per rule (mask/redact/hash/encrypt/generalize/suppress).
6. Execute transform and export protected dataset.

7.3 SynthGen workflow

1. Open `SynthGen`.
2. Select source dataset and generation profile.
3. Configure sample size and optional constraints.
4. Start generation job.
5. Download synthetic dataset artifact on completion.

7.4 Data quality and parity checks

1. Schema parity: - Column count and names are as expected.
2. Privacy parity: - Original sensitive values are not exposed in transformed output.
3. Utility parity: - Basic distribution trends remain usable for modeling intent.
4. Integrity: - Output file is readable and row count matches configured generation target.

7.5 Functional validation checklist

1. Privacy detection returns non-empty findings when PII exists.
2. Transform run produces downloadable artifact.
3. SynthGen run reaches terminal `completed` state.
4. Generated dataset can be loaded into Datasets/Experiments modules.
5. Error states surface actionable diagnostics.

7.6 Expected result

1. Sensitive data is protected according to selected policy.
2. Synthetic data can be used for prototyping and tests.

7.7 Common errors and recovery

1. No PII findings when expected: - Verify selected policy/rules and source columns.
2. Transform failure: - Retry with narrower rule scope and inspect logs.
3. SynthGen timeout: - Reduce generation size and rerun.

7.8 Screenshots

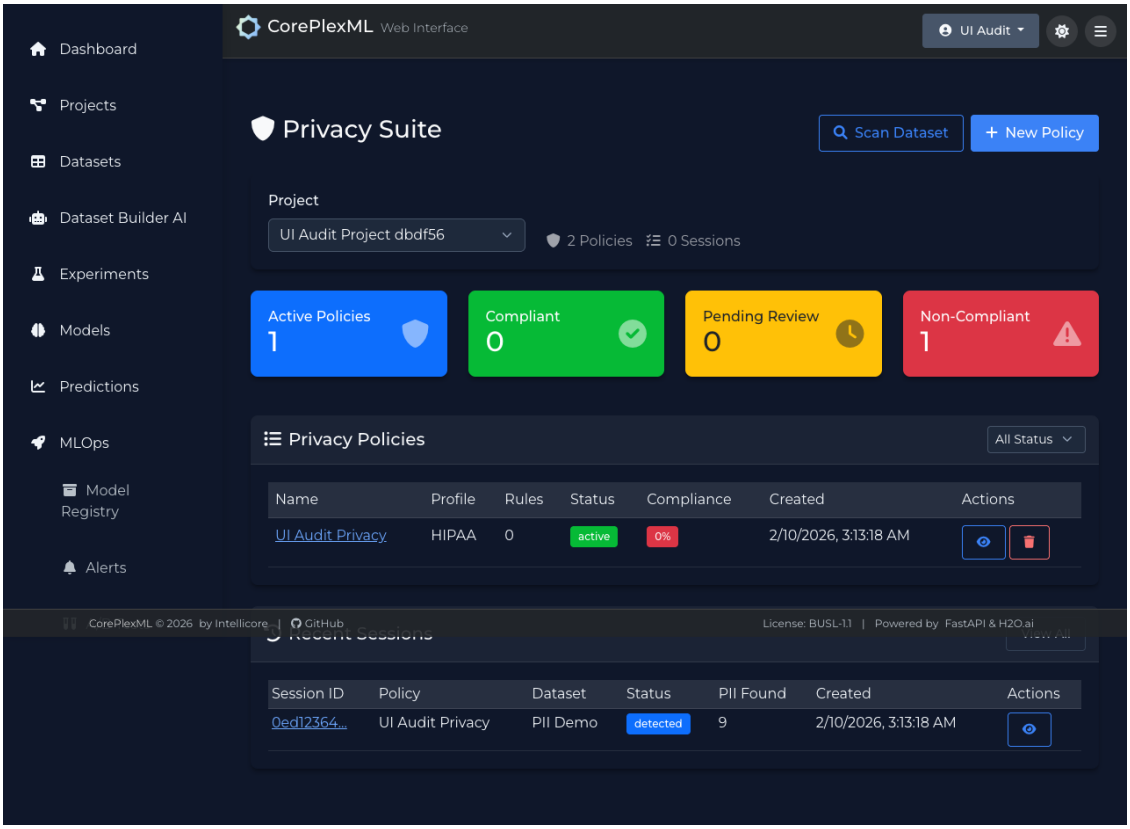


Fig. 1: Privacy module with policy execution flow.

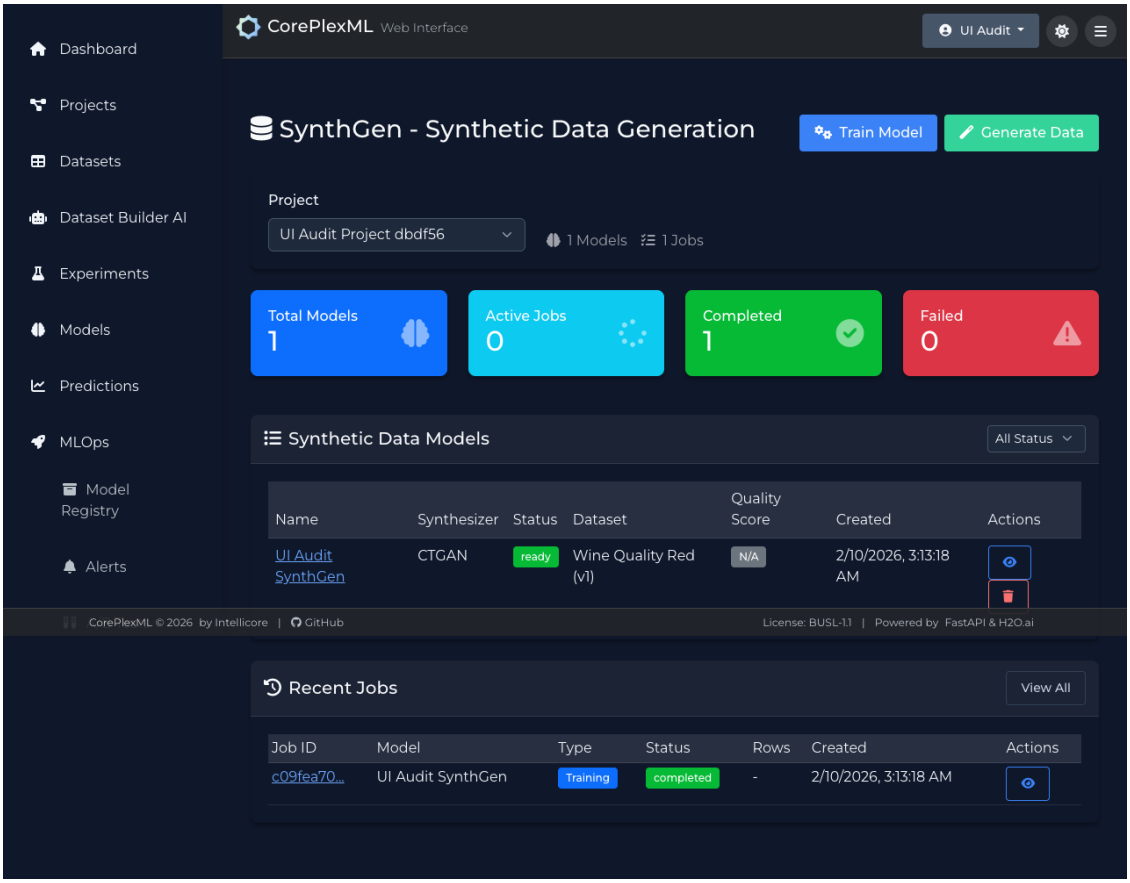


Fig. 2: SynthGen training/generation and artifact output view.

CHAPTER

EIGHT

8. TEAMS, BILLING, ADMIN, TROUBLESHOOTING

8.1 Goal

Operate team collaboration, subscription lifecycle, and admin controls with predictable outcomes.

8.2 Teams management

1. Open `Teams`.
2. Create team or open existing team workspace.
3. Invite members by email.
4. Assign role per member (owner/admin/member according to policy).
5. Validate invited user can access only authorized modules.

8.3 Billing and plan lifecycle

1. Open profile/billing section.
2. Validate current plan state (Free, Pro, Team).
3. Execute upgrade flow when needed: - Free -> Pro. - Pro -> Team.
4. Confirm billing status updates after successful checkout.
5. Verify team entitlements are unlocked for Team plan.

8.4 Admin controls

1. Open `Admin` (authorized roles only).
2. Review: - Audit logs. - Operational status pages. - Tenant/team controls.
3. Apply changes only with change ticket or approval workflow.

8.5 Functional validation checklist

1. Team invite, acceptance, and role assignment complete successfully.
2. Plan upgrades reflect in UI and permission model.
3. Subscription status survives logout/login and refresh.

4. Admin pages are inaccessible to unauthorized roles.
5. Billing errors are explicit and do not leave ambiguous plan state.

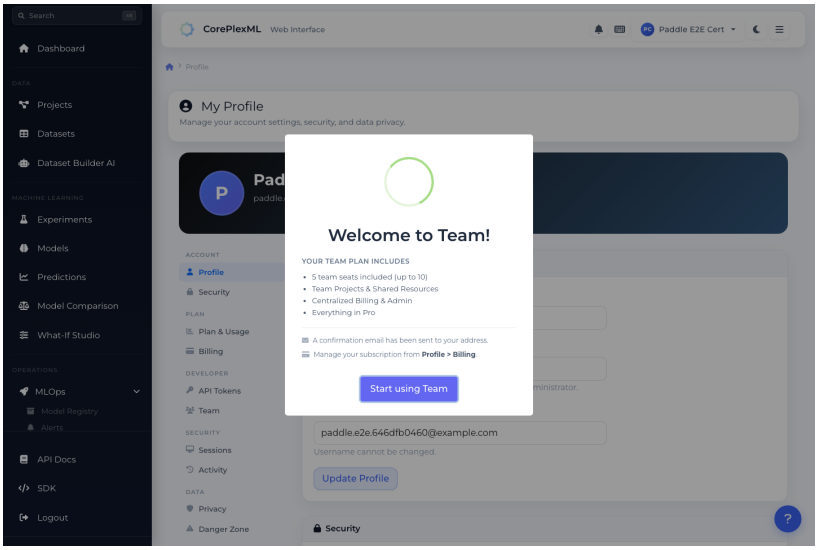
8.6 Expected result

1. Collaboration and plan controls work end-to-end.
2. Operational admins can diagnose issues without data corruption.

8.7 Troubleshooting quick playbook

1. Session/auth issue: - Logout/login and validate account state.
2. Missing module/data: - Validate active project and role permissions.
3. Billing mismatch: - Verify subscription provider status and team ownership.
4. UI stale state: - Hard refresh and rerun last action once.

8.8 Screenshot



Team and billing controls from profile/workspace context.