Technical Design Document (TDD)

Structuring Databricks for Success: Folders, Access, Governance

Contents

1. Purpose

This document outlines the best practices for organizing folders and artifacts in Databricks to ensure clarity, scalability, collaboration, and governance across teams and environments. It is tailored for the implementation of a project: **SBK Customer360 Platform**.

2. Scope

Applies to all Databricks workspaces used across data engineering, analytics, and ML teams, supporting structured workflows in development, testing, and production environments. This structure supports SBK Customer360 Platform which consolidates sales, support, marketing, and digital footprint data into a unified customer profile.

3. Assumptions

- Unity Catalog is enabled for governance.
- Git integration is in place.
- CI/CD workflows are adopted for deployments.
- Customer360 uses Azure Databricks with enterprise-level workspace governance.

4. Folder Structure Design

4.1 Top-Level Structure

```
/Shared
  └── Customer360
        ├── dev/
        ├── test/
        └── prod/
```

Dev/	Development workspace for experimentation, unit testing, and sandbox
Test/	Environment for integration testing and QA validation workflows
Prod/	Production-ready notebooks, jobs, and pipelines for live execution or Alternative (Team-Based)

```
/Shared
   ├── customer360-data-engineering/ #Contains ingestion scripts, ETL notebooks, DLT pipeline
   ├── customer360-analytics/        # Dashboards, reporting queries, view
   └── customer360-mlops/            # ML models, training workflow, Mlflow, deployment
```

4.2 Subfolders (per Environment)

```
/Shared/Customer360/dev/
   ├── notebooks/            # Contains exploration, transformation, or EDA notebooks
   ├── jobs/                 # Databricks job definitions and scheduled tasks
   ├── pipelines/            # Delta Live Tables or structured ETL pipelines
   ├── dbt_models/           # DBT models and configs if using dbt-core
   ├── tests/                # Data quality and unit/integration tests
   └── docs/                 # Markdown files, architecture diagrams, README
```

5. Naming Conventions

Artifact Type	Naming Convention Example
Notebook	01_ingest_crm_data, 02_transform_support_tickets
Job	load_web_events_dev
Pipeline	curate_customer_gold_profile_dlt
Table	customer_bronze, customer_silver, customer_gold
Test Script	test_missing_values_crm_customers

6. Governance with Unity Catalog

- Catalogs: `sbk_customer360`
- Schemas: `bronze`, `silver`, `gold`
- Access Control: Role-based via Unity Catalog (e.g., marketing_read, eng_write)
- Sensitive Data: Apply row/column-level access policies for fields like `email`, `phone`, `ssn`

Governance with Unity Catalog
1. Create Catalog and Schemas

```
sql
CopyEdit
-- Create catalog
CREATE CATALOG IF NOT EXISTS sbk_customer360
COMMENT 'Customer 360 catalog for all customer-related data (ACME brand)';

-- Set owner
ALTER CATALOG sbk_customer360 OWNER TO `accountadmin`;

-- Create schemas
CREATE SCHEMA IF NOT EXISTS sbk_customer360.bronze COMMENT 'Raw landing
data';
CREATE SCHEMA IF NOT EXISTS sbk_customer360.silver COMMENT 'Cleaned and
transformed data';
CREATE SCHEMA IF NOT EXISTS sbk_customer360.gold COMMENT 'Curated data for
analytics';

 2. Grant Role-Based Access Control (RBAC)
sql
CopyEdit
-- Grant read-only access to marketing
GRANT USAGE ON CATALOG sbk_customer360 TO `marketing_read`;
GRANT SELECT ON ALL TABLES IN SCHEMA sbk_customer360.gold TO
`marketing_read`;

-- Grant full access to engineers
GRANT USAGE ON CATALOG sbk_customer360 TO `eng_write`;
GRANT CREATE, SELECT, MODIFY ON SCHEMA sbk_customer360.silver TO `eng_write`;

-- Optional: Auto-grant on future objects
GRANT SELECT ON FUTURE TABLES IN SCHEMA sbk_customer360.gold TO
`marketing_read`;

3. Define and Apply Row Access Policy
 Goal: Restrict rows by region or department
sql
```

```
CopyEdit
-- Create Row Access Policy
CREATE OR REPLACE ROW ACCESS POLICY region_policy
AS (region STRING)
USING (CURRENT_USER() IN ('user1@example.com', 'user2@example.com') OR region =
'EMEA');



 Apply to table
sql
CopyEdit
ALTER TABLE sbk_customer360.gold.customer_profile
ADD ROW ACCESS POLICY region_policy ON (region);

 4. Define and Apply Column Masking for Sensitive Fields
 Masking Policy for Email
sql
CopyEdit
CREATE OR REPLACE MASKING POLICY email_mask
AS (email STRING)
USING (CASE
         WHEN is_account_group_member('pii_access') THEN email
         ELSE '***MASKED***'
       END);
 Masking Policy for SSN
sql
CopyEdit
CREATE OR REPLACE MASKING POLICY ssn_mask
AS (ssn STRING)
USING (CASE
         WHEN is_account_group_member('pii_access') THEN ssn
         ELSE NULL
       END);
Apply to table columns
sql
CopyEdit
```

```
ALTER TABLE sbk_customer360.gold.customer_profile
ALTER COLUMN email
SET MASKING POLICY email_mask;

ALTER TABLE sbk_customer360.gold.customer_profile
ALTER COLUMN ssn
SET MASKING POLICY ssn_mask;

Optional: Create Access Groups
In the admin console (or SCIM APIs), create the following access groups:
  • marketing_read
  • eng_write
  • pii_access
Then assign users to these groups accordingly.
```

7. Git Integration & CI/CD

Repo Structure:

```
/customer360-repo
  ├── src/
  │     ├── notebooks/
  │     ├── workflows/
  │     └── utils/
  ├── tests/
  ├── docs/
  └── .github/
        └── workflows/
```

Versioning Format: `.py` for production workflows, `.ipynb` for exploration.

/customer360-repo Folder Structure

Folder Path	Purpose
src/	Source code directory for all core data workflows and assets
└── **notebooks/**	Reusable notebooks for ingestion, transformation, and exploration
└── **workflows/**	Production-ready pipeline scripts (DLT, batch jobs, streaming)
└── **utils/**	Shared utility functions and helper modules (e.g., logging, schema defs)
tests/	Unit tests and data validation scripts for CI pipelines
docs/	Architecture diagrams, onboarding notes, and markdown documentation
.github/	GitHub-specific folder for CI/CD automation
└── **workflows/**	YAML pipelines for Databricks job deployment, DBT builds, code checks

8. Job Management

- Use Tags: `env: dev`, `project: customer360`
- Naming: `load_customer_demographics_prod`, `sync_support_data_test`
- Folder Grouping: Jobs organized under `/jobs/` per environment with alert policies configured

Job Management Standards

Practice	Description
Use Tags	Assign tags such as env: dev, env: prod, and project: customer360 to all jobs for better filtering, traceability, and automation triggers.
Naming	Use descriptive, lowercase names with underscores for clarity. Example job names: load_customer_demographics_prod, sync_support_data_test.
Folder Grouping	Organize jobs under environment-specific folders (e.g., /jobs/dev/, /jobs/prod/) and configure alert policies for monitoring SLA violations or failures.

This helps ensure:

- Easier job search and filtering in the UI
- Cleaner DevOps workflows with consistent identifiers
- Better observability via alerts and notifications

9. Delta Live Tables (DLT)

```
/Shared/Customer360/prod/pipelines/
  ├── dlt_ingest_web_events.py
  ├── dlt_curate_crm.py
  └── dlt_build_customer_gold.py
```

Use `@dlt.table`, `@dlt.expect` decorators for schema enforcement and SLAs.

Delta Live Tables (DLT) Pipeline Structure

File Path	Purpose
dlt_ingest_web_events.py	Ingests raw website event data from tracking systems into the bronze layer using @dlt.table decorators.
dlt_curate_crm.py	Applies transformation logic to CRM records, filters invalid rows, and writes curated data to the silver layer.

dlt_build_customer_gold.py	Joins CRM, support, and web data to produce a unified customer profile in the gold layer for analytics.

This layout enables:

- Modular pipeline development by domain/stage
- Easier monitoring and SLA enforcement
- Reusability across environments via parameterized configs

10. Permissions and Access Control

- Folder-level and notebook-level ACLs per team
- Unity Catalog governance via group policies
- RBAC for roles like `marketing_analyst`, `data_engineer`, `auditor`

Permissions and Access Control

Control Mechanism	Description
Folder & Notebook ACLs	Set **folder-level and notebook-level access controls** to restrict visibility and edit rights by team or individual contributor.
Unity Catalog Group Policies	Leverage **Unity Catalog governance** by applying group-based permissions for catalogs, schemas, and tables (e.g., marketing_read, eng_write).
Role-Based Access Control (RBAC)	Assign predefined roles like marketing_analyst, data_engineer, and auditor to enforce fine-grained access across data layers and tools.

This ensures:

- Strong data security across dev, test, and prod environments
- Seamless collaboration without permission conflicts
- Auditability aligned with enterprise governance standards

11. Cleanup & Retention

- Delta table retention: 30 days for bronze, 90 days for silver
- Notebooks archived to Git monthly
- Job results and logs auto-expired after 14 days using workspace settings

Clean UP

1. Delta Table Retention Policy

Set **Time Travel retention** using ALTER TABLE ... SET TBLPROPERTIES.

Bronze → 30 Days

sql

CopyEdit

```
ALTER TABLE sbk_customer360.bronze.web_events
SET TBLPROPERTIES ('delta.deletedFileRetentionDuration' = '30 days');
```

Silver → 90 Days

sql

CopyEdit

```
ALTER TABLE sbk_customer360.silver.crm_clean
SET TBLPROPERTIES ('delta.deletedFileRetentionDuration' = '90 days');
```

This controls how long **old versions and deleted files** are kept for Time Travel and vacuum safety.

2. Notebook Archival to Git (Monthly)

Automate via CI/CD or a scheduled GitHub Action.

🔧 **Sample GitHub Workflow (.github/workflows/archive_notebooks.yml)**

yaml

CopyEdit

```
name: Archive Notebooks Monthly
```

```
on:
  schedule:
    - cron: '0 1 1 * *'  # Runs on the 1st day of every month at 1 AM
  workflow_dispatch:

jobs:
  archive:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout notebooks
        uses: actions/checkout@v3

      - name: Commit archived notebooks
        run: |
          git config user.name "databricks-bot"
          git config user.email "bot@databricks.com"
          git add src/notebooks/
          git commit -m "Monthly archive of notebooks - $(date +'%Y-%m-%d')" ||
echo "No changes"
          git push origin main
 Ensure notebooks are in .py or .dbc format via %notebook save or Repos sync.
```

3. Job Results and Log Expiry – 14 Days

This is configured in **Databricks Admin Console**, but here's how to set it via **workspace APIs** or Terraform.

Terraform (if using automation)

```
hcl
CopyEdit
resource "databricks_workspace_conf" "job_retention" {
  custom_config = {
    "spark.databricks.jobResultsRetentionInDays" = "14"
  }
}
```

Or manually go to **Admin Console → Workspace Settings** → set:

- Job Results Retention = **14 days**

Summary Table

Item	Code / Action
Delta Bronze Retention (30 days)	ALTER TABLE ... SET TBLPROPERTIES
Delta Silver Retention (90 days)	ALTER TABLE ... SET TBLPROPERTIES
Notebook Archival to Git	GitHub Actions cron job
Job Logs Expiry (14 days)	Admin Console or databricks_workspace_conf via Terraform

12. Documentation Standards

- Markdown files in `/docs/`
- Content:
  - System overview (Customer360 architecture)
  - Customer entity schema
  - Job ownership and contact
  - External data contracts (CRM, Support API, Web tracking)

Documentation Standards

Location	Description
/docs/	Centralized location for project documentation in Markdown format (.md).

System Overview	Includes high-level architecture of the Customer360 platform, data flow, and environment separation.
Entity Schemas	Defines structure, keys, and relationships for core entities (e.g., customers, transactions, interactions).
Job Ownership	Maps each notebook or job to an owner with email/contact and escalation paths.
Data Contracts	Lists external dependencies such as CRM systems, support APIs, and web tracking inputs, including refresh frequency and contact points.

This ensures:

- Easy onboarding for new engineers and analysts
- Clear accountability and traceability
- Alignment with upstream and downstream data providers

13. Tools & Integrations

Tool	Purpose
Unity Catalog	Access control and data cataloging
GitHub	Version control and automated CI/CD
MLflow	Track experiments for churn model
Terraform	Automate workspace and cluster setup
Azure Monitor	Monitor job performance and SLA violations

14. Summary

This folder structure and governance model supports Customer360 initiative by enabling scalable, secure, and maintainable workflows across data ingestion, transformation, and analytics. It provides unified access control, automation readiness, and data transparency for all customer data stakeholders.

Final Note

This document serves as a **technical blueprint** for organizing Databricks workspaces in a scalable, secure, and production-ready manner—tailored to the needs of the **Customer360 initiative at SBK**.

It is intended for use by:

- **Data Engineers** – to follow consistent folder and artifact conventions across environments
- **Analytics & ML Teams** – to integrate cleanly with curated layers, governed access, and reusable workflows
- **Platform Engineers** – to enforce workspace hygiene, CI/CD pipelines, Unity Catalog policies, and automation
- **Solution Architects & Project Leads** – to ensure the workspace aligns with enterprise governance, performance, and compliance needs

By following this structure, teams across SBK can accelerate development, improve operational transparency, and ensure trust in data across all business functions.

Appendix:

How This Template Helps

Category	How the Template Helps
Folder Organization	Provides a **repeatable and scalable folder structure** across dev/test/prod.
Artifact Naming	Avoids ambiguity by standardizing names for jobs, notebooks, tables, and pipelines.
Governance	Integrates Unity Catalog for **RBAC**, sensitive data masking, and schema control.
CI/CD	Lays out Git repo layout and versioning practices for seamless **collaborative dev**.
DLT and Pipelines	Prepares teams for building **Delta Live Table pipelines** with quality expectations.
Job and Resource Management	Includes best practices for **job tagging**, folder separation, and cleanup retention.
Documentation Standards	Promotes clear architecture diagrams, data dictionaries, and team accountability.
Infrastructure as Code	Prepares you for automating the environment using **Terraform or similar tools**.

Thank You Note

Thank you for taking the time to explore this technical design document. We hope it provides clarity, structure, and actionable insights for building scalable, governed, and collaborative workflows in Databricks.

Whether you're a data engineer, architect, analyst, or platform lead, your role is critical in shaping reliable and future-ready data systems.
Let's keep building with clarity, security, and purpose.

Feel free to share feedback, ideas, or improvements — because great architecture grows through collaboration.