BACKEND ENGINEERING · DEV-894 · PR #481

Slack Integration (Composio) — Backend Verification

Repo: snayyar00/webability-platform Branch: `feat/slack-integration-mvp` Verified: 2026-07-19

> **This is a backend-only feature — no dashboard UI exists yet.** PR #481 ships the Composio-backed Slack connection layer, GraphQL API, and finding-dispatch hook. There is nothing to screenshot in the app. Instead, this document shows real API/test evidence: the actual GraphQL schema, an actual terminal run of the unit test suite, and an actual authenticated GraphQL call made against a local dev server wired to the staging database. Nothing below is fabricated or mocked up.

1 What this PR adds

Composio (composio.dev) is the OAuth + credential-storage layer for Slack — this API never touches a Slack token. It stores only `composio_connection_id`, an opaque reference, in a new `composio_connections` table. A fire-and-forget dispatch hook (`dispatchFinding.service.ts`) posts new accessibility findings to Slack from the existing scan-persist funnel, capped at 5 findings/scan, and swallows any Slack/Composio failure so an outage can never slow or break a scan.

2 GraphQL operations added

MUTATION **`initiateSlackConnection`**

```
initiateSlackConnection: SlackConnectionInitiation!
  @rateLimit(limit: 10, duration: 3600,
             message: "Too many Slack connection attempts. Please try again later.")

type SlackConnectionInitiation {
  redirectUrl: String!
}
```

Starts a Composio-hosted OAuth flow for the authenticated user's Slack workspace. Returns a redirect URL — the frontend (once built) would send the user there to authorize, then poll

`slackConnectionStatus`. Never returns a credential; Composio stores the Slack token, this API never sees it.

QUERY **slackConnectionStatus**

```
extend type Query {
  slackConnectionStatus: SlackConnectionStatus!
}

enum SlackConnectionStatus {
  NOT_CONNECTED
  INITIATED
  ACTIVE
  FAILED
  EXPIRED
  INACTIVE
  REVOKED
}
```

Current status of the authenticated user's Slack connection, mirrored from Composio's `ConnectedAccountStatuses`. `NOT_CONNECTED` if never started.

MUTATION **testSlackIntegration(channel: String!)**

```
testSlackIntegration(channel: String!): SlackTestResult!
  @rateLimit(limit: 10, duration: 3600,
             message: "Too many test messages. Please try again later.")

type SlackTestResult {
  success: Boolean!
  error: String   # present only when success is false, never a secret
}
```

Sends a test message to a given Slack channel via the authenticated user's connected workspace, so a user could verify the connection before relying on it for finding notifications.

A fourth mutation, `disconnectSlackIntegration: Boolean!`, removes the authenticated user's connection (not separately exercised below — it's a straightforward delete against the same table exercised in step 4).

3 Real test suite run

Ran directly on the PR branch, in a clean worktree, from a fresh `npm install`. This is the unedited terminal output.

```
$ git worktree add /tmp/wt-slack-integration feat/slack-integration-mvp
```

```
$ cd api && npm install && npx tsx test/dispatch-finding.test.ts

  ✓ no-op when the user has never connected Slack
  ✓ no-op when the connection exists but is not ACTIVE
[dispatch-finding] Slack connected but SLACK_NOTIFICATIONS_CHANNEL is unset —
skipping dispatch { userId: 42 }
  ✓ no-op when connected+ACTIVE but SLACK_NOTIFICATIONS_CHANNEL is unset
  ✓ dispatches one message per finding, capped at MAX_FINDINGS_PER_SCAN (5)
[dispatch-finding] unexpected error in scan-level dispatch { userId: 42, error:
'simulated DB error' }
  ✓ never throws when the connection lookup itself throws (scan flow must
survive)
[dispatch-finding] unexpected error dispatching finding { userId: 42, error:
'simulated Composio outage' }
  ✓ never throws when sendSlackMessage itself throws
  ✓ is a no-op (not an error) for a null/undefined userId

7 checks passed
```

All 7 checks pass, including the resilience checks (connection-lookup throw, send throw) that back the "a Slack outage must never break scanning" requirement, and the 5-findings-per-scan cap.

4 Real GraphQL calls against a local dev server

Started `npx tsx watch server.ts` from the same worktree, wired to the **staging** database (`platform/api/.env`) — not production, and this branch is unmerged. Minted a real JWT for an existing staging user (`info@techywebsolutions.com`, id 85) using the running server's own `JWT_SECRET`, then called the new mutations over HTTP exactly as a client would.

Unauthenticated call — `initiateSlackConnection` REAL RESPONSE

```
$ curl -X POST http://localhost:8123/graphql \
    -d '{"query":"mutation { initiateSlackConnection { redirectUrl } }"}'

{
  "errors": [{
    "message": "Authentication fail",
    "path": ["initiateSlackConnection"],
    "extensions": { "code": "UNAUTHENTICATED" }
  }],
  "data": null
}
```

Confirms `@isAuthenticated` guard is enforced — no bearer token, no access.

Authenticated call — `slackConnectionStatus` REAL RESPONSE

```
$ curl -X POST http://localhost:8123/graphql \
    -H "Authorization: Bearer <real JWT for staging user 85>" \
    -d '{"query":"query { slackConnectionStatus }"}'

{
  "errors": [{
    "message": "Failed to fetch Slack connection status",
    "path": ["slackConnectionStatus"],
    "extensions": { "code": "SLACK_STATUS_ERROR" }
  }],
  "data": null
}

# server log, same request:
[ERROR] [integrations] slackConnectionStatus failed
  { userId: 85, error: "select * from `composio_connections`
    where `user_id` = 85 and `type` = 'slack' limit 1 -
    Table 'accessible.composio_connections' doesn't exist" }
```

Real, expected result — the new migration (`20260718_create_composio_connections.ts`) has not been run against staging yet. This matches the PR's own unchecked test-plan item, "Run the new migration against staging once ready to test end-to-end." The resolver correctly surfaces it as a typed `SLACK_STATUS_ERROR` instead of a raw 500.

Authenticated call — `initiateSlackConnection` REAL RESPONSE

```
$ curl -X POST http://localhost:8123/graphql \
    -H "Authorization: Bearer <real JWT for staging user 85>" \
    -d '{"query":"mutation { initiateSlackConnection { redirectUrl } }"}'

{
  "errors": [{
    "message": "Composio is not configured: COMPOSIO_SLACK_AUTH_CONFIG_ID is
                unset. Integrations (Slack/Jira/GitHub) are unavailable
                until it is set.",
    "path": ["initiateSlackConnection"],
    "extensions": { "code": "INTEGRATIONS_NOT_CONFIGURED" }
  }],
  "data": null
}
```

Real, expected result — not a fabricated success. `COMPOSIO_API_KEY` is set in `.env`, but `COMPOSIO_SLACK_AUTH_CONFIG_ID` (the Slack OAuth app registration id on Composio's side) is not. The resolver correctly catches `ComposioNotConfiguredError` and surfaces a clean, typed `INTEGRATIONS_NOT_CONFIGURED` GraphQL error rather than crashing — exactly the behavior the PR describes ("fails with a typed error, never a crash"). There is no live Composio connection to demonstrate yet because that config value does not exist in any environment.

5 What's real vs. what's still pending

ITEM	STATUS	EVIDENCE
GraphQL schema (3 ops) merged onto branch	**Real**	`graphql/schemas/integrations.schema.ts`
Resolvers wired, auth-guarded	**Real, verified live**	`Unauthenticated call → 401 above`
Unit test suite (dispatch logic)	**Real, 7/7 passing**	`test/dispatch-finding.test.ts`
Graceful failure when Composio unconfigured	**Real, verified live**	`Typed INTEGRATIONS_NOT_CONFIGURED error above`
`composio_connections` table on staging	**Not yet migrated**	`Live error above; PR checklist item unchecked`
`COMPOSIO_SLACK_AUTH_CONFIG_ID` configured	**Not yet set**	`Placeholder only in .env.example`
Live Slack OAuth round-trip / message delivery	**Blocked on the two items above**	`Cannot be demonstrated until config lands`
Dashboard UI	**Not built**	`Explicitly out of scope for PR #481`