Admin Portal Dec-2025 Releases

1. API Stubbing Module (New)

1.1 Introduction

The **API Stubbing module** lets you simulate APIs for testing. It enables you to create responses without calling real services. It's really helpful for testing parts of your software, making development faster, and keeping your tests reliable.

This new version gives you an easy-to-use screen to create, change, and run these simulated API responses. You get lots of control over your testing. It offers three main types of stubs: **Full Static**, **Conditional**, and **Passthrough + Override**. You can also fine-tune the **Response Body, Headers, Status, and how it behaves**.

This document will walk you through the new features, how to use them, and some tips to help you get started quickly. Your team can use these powerful new tools right away.

> ⓘ This feature makes API stubbing through Admin Portal much easier to use and understand.

1.2 How to Find the Module

Finding the new API Stubbing module is simple:

- Login into admin portal
- Go to the dashboard: **Tools Section → API Stubbing**.
- The **Add API Stubbing** button is easy to find at the top of the list page. Click it to start setting up new stubs quickly.
- You can search for, change, or delete your existing stubs from the main list. It's your central place for all stub tasks.

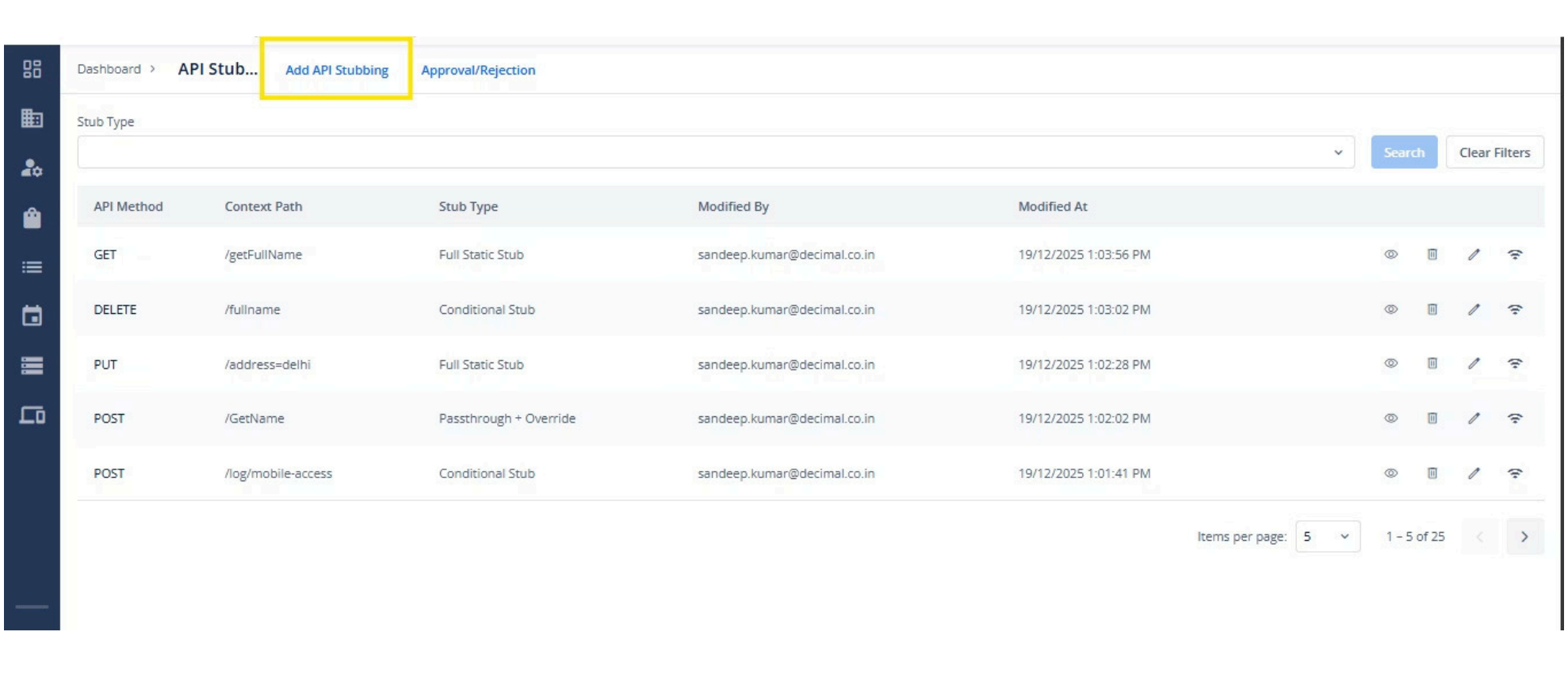

Fig 1 : [Screenshot: API Stubbing Listing Page. See how you can easily Add, Edit, or Delete stubs]

1.3 Key Features

This release comes with powerful tools to make your API testing and development better:

- **"Add API Stubbing" Button**: A quick button to start making new stubs directly at the top section of the list in the list screen.
- **API Method Selection**: Choose common actions like GET, POST, PUT, DELETE. This helps you test different kinds of API requests.
- **Auto-generated URL Path Pattern**: The system automatically creates URL patterns for you. You can also add an optional **Context Path** to make your API addresses more specific.
- **Flexible Stub Types**:
  - **Full Static Stub**: For simple, unchanging responses every time. Great for basic tests that need a fixed answer.
  - **Conditional Stub**: Gives different responses based on certain rules or conditions in the request. You can set detailed rules with our Business Rule Engine for advanced testing.
  - **Passthrough + Override**: Sends requests to the real API, but lets you change parts of the response before it returns to your application. Useful for testing errors or specific data changes without affecting the live service.
- **Rule Card Management**: For Conditional Stubs, you can set up many rules. Each rule has a unique **Rule Name**, its own **Response**, and a specific search rule (**Search BRE**) to match incoming requests.
- **Response Body Editor**: A smart editor that checks your JSON code as you type. It also supports templates to create dynamic content (like using information from the request). Plus, you can easily copy and format your JSON to make it readable.
- **Response Headers & Status**: Gives you control over extra information sent with the response. You can set custom headers (like Content-Type) and specific HTTP status codes (like 200 OK for success or 404 Not Found for an error).
- **Behavior Simulation**: Test how your app handles real-world problems. Add **Artificial Delay** to pretend there's slow internet, or use network fault simulation to mimic server issues. This helps build stronger applications.
- **Stub Management**: Easily move, export, import, back up, and restore your stub settings. This makes sure your stubs are easy to share and always safe.

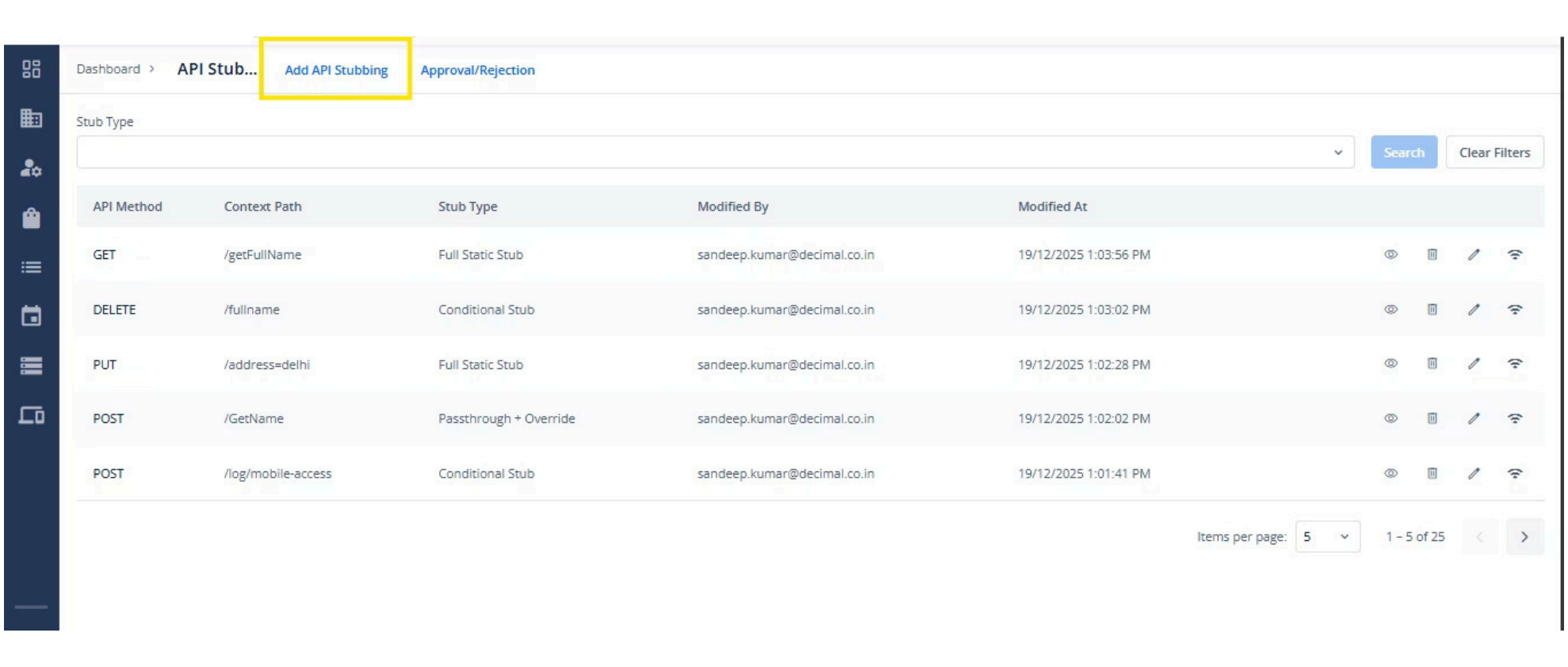

Fig 1 : [Screenshot: API Stubbing - Add API Stub Page. This shows the clear layout for making new stubs]

1.4 Creating a Stub: Easy Steps

Follow these steps to create a new API stub:

Step 1: Basic Information

- Choose the **API Method**: Pick GET, POST, PUT, or DELETE. This tells the stub how to react to different types of requests.
- Check the **URL Path Pattern**: The URL Path Pattern is automatically generated based on your selections.
- Add an optional **Context Path**: Use this to add more to your API address, like a version number (e.g., /api/v1) or to separate different services or to add the similar context path as your original API.

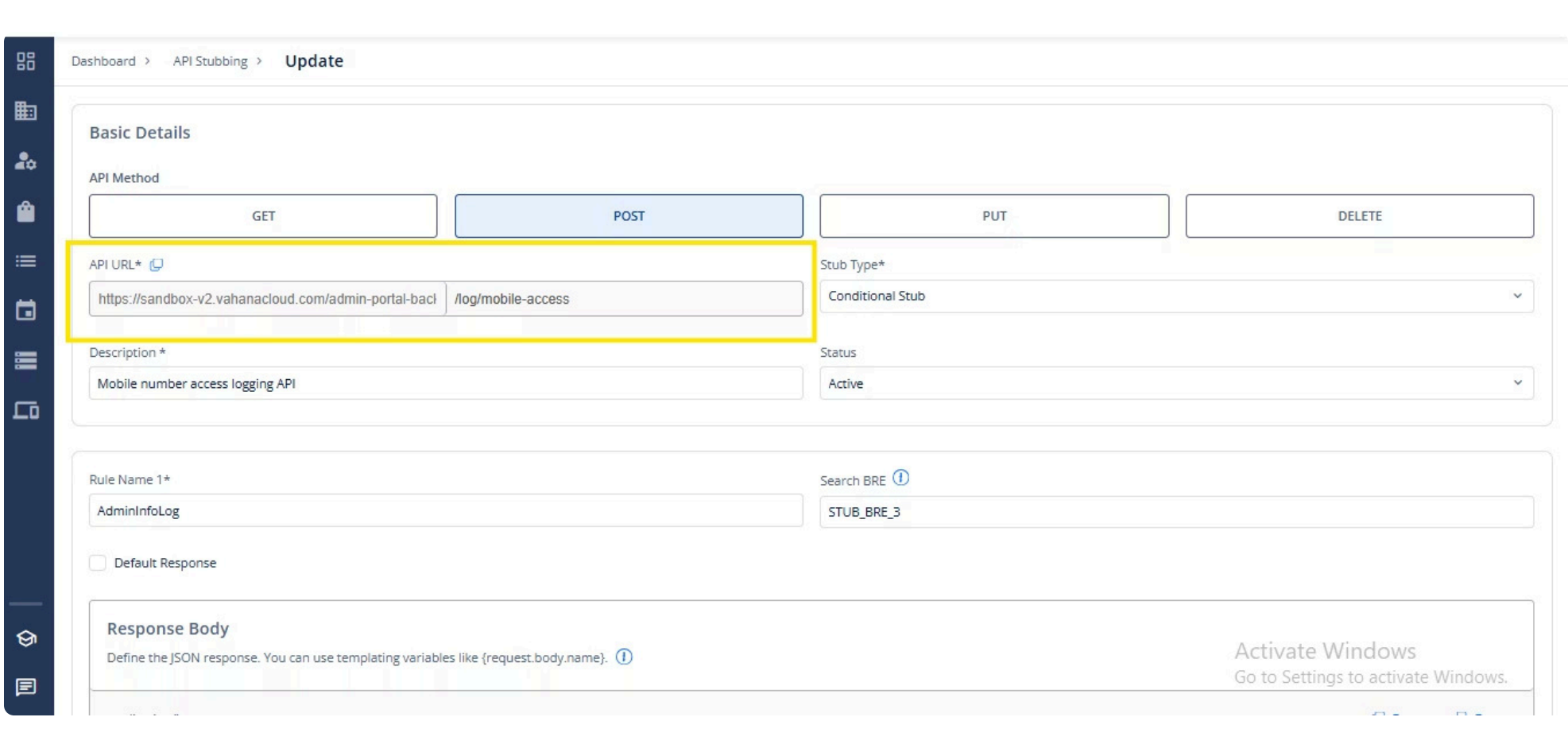

Fig 3 : [Screenshot: User can copy the API URL along with the Context path]

Step 2: Pick Your Stub Type

Choose the best stub type for your test:

- **Full Static Stub** – Use this when you need the exact same response every time. Perfect for simple tests.
- **Conditional Stub** – Great for testing complex situations. You can set rules to get different responses based on things in the incoming request (like specific headers or body content).
- **Passthrough + Override** – This advanced option sends the request to the real API but lets you change its response before it gets back to your app. It's good for testing errors or making small changes without touching the live service.

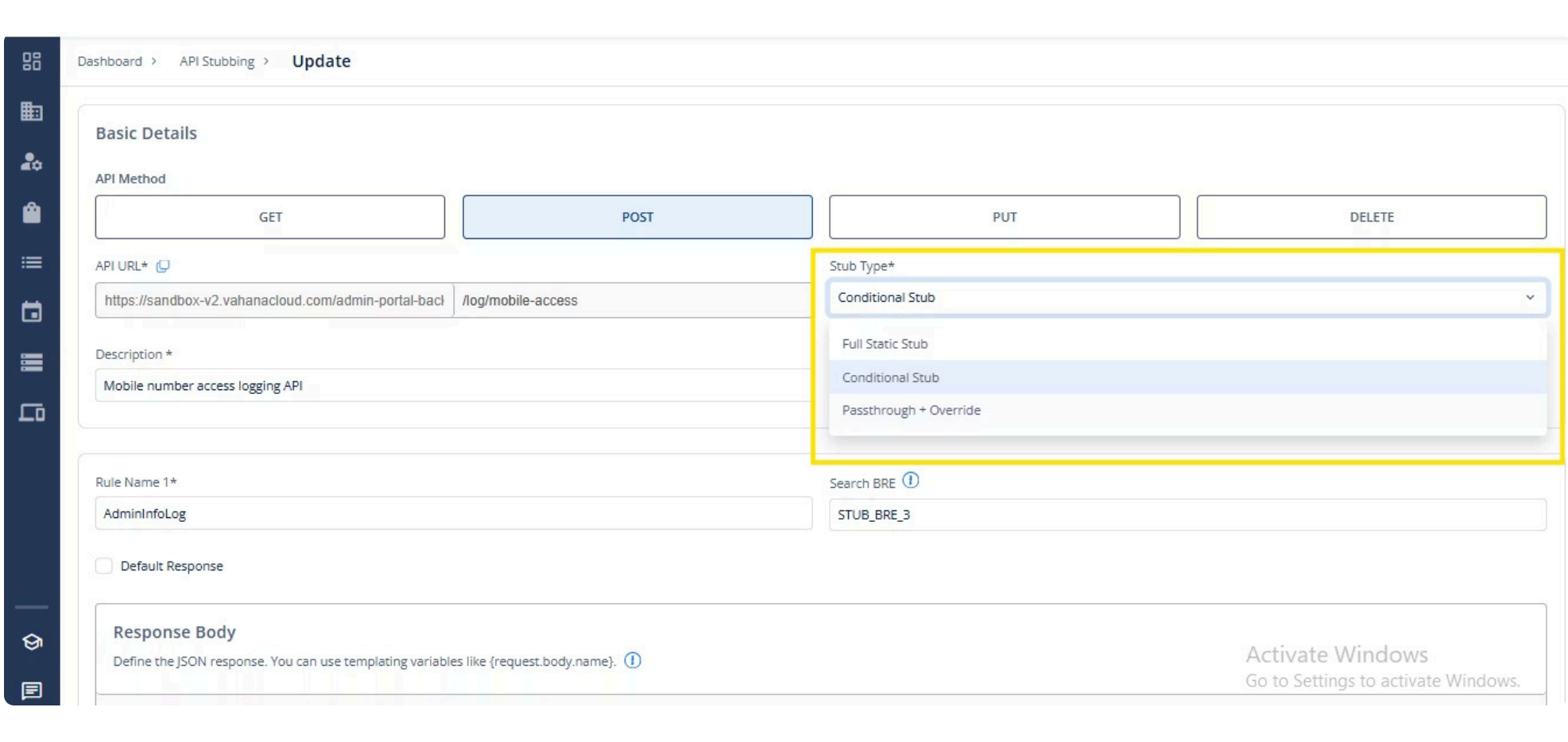

Fig 4 : [Screenshot: Type of Stubbing : Full Static Stub, Conditional Stub, and Passthrough + Override]

Step 3: Set Up Rules (If You Chose Conditional)

If you're using Conditional Stubs, setting rules is important:

- Add several rules. Each needs a unique **Rule Name**, a specific **Response** (what the body, headers, and status should be), and a **Search BRE** (Business Rule Engine) to decide when this rule's response should be used.
- Rules are checked from top to bottom. Make sure your most specific rules are at the top.

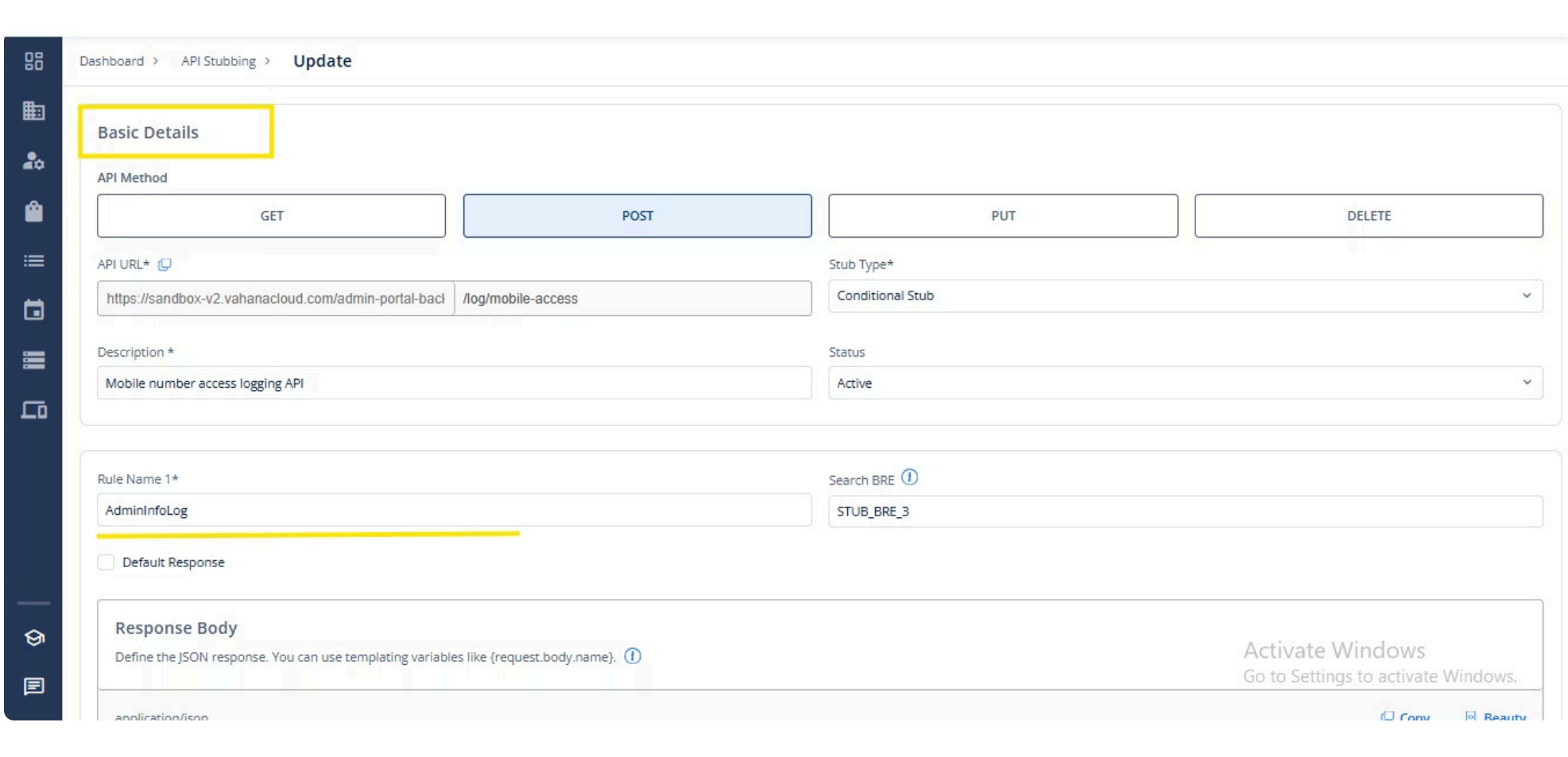

Fig 5 : [Screenshot: Rule Card "Add Rule" Button. Shows how to set up responses that change based on conditions.]

Step 4: Response Body

Decide what data your stub will send back:

- Type your JSON data into the **Response Body** field. This can be a simple message or a detailed data structure.
- Use **templating variables** (like {request.body.name} or {request.headers.token}) to make responses dynamic, pulling info from the original request.
- Handy buttons let you **Copy** the response or **Beautify JSON** to make it easier to read.

Step 5: Response Headers & Status

Control the extra technical details of your fake response:

- Set the **HTTP Status Code** (e.g., 200 for success, 400 for a bad request, 500 for a server error). This helps simulate different API outcomes.
- Add any **custom headers** you need, such as Content-Type: application/json.

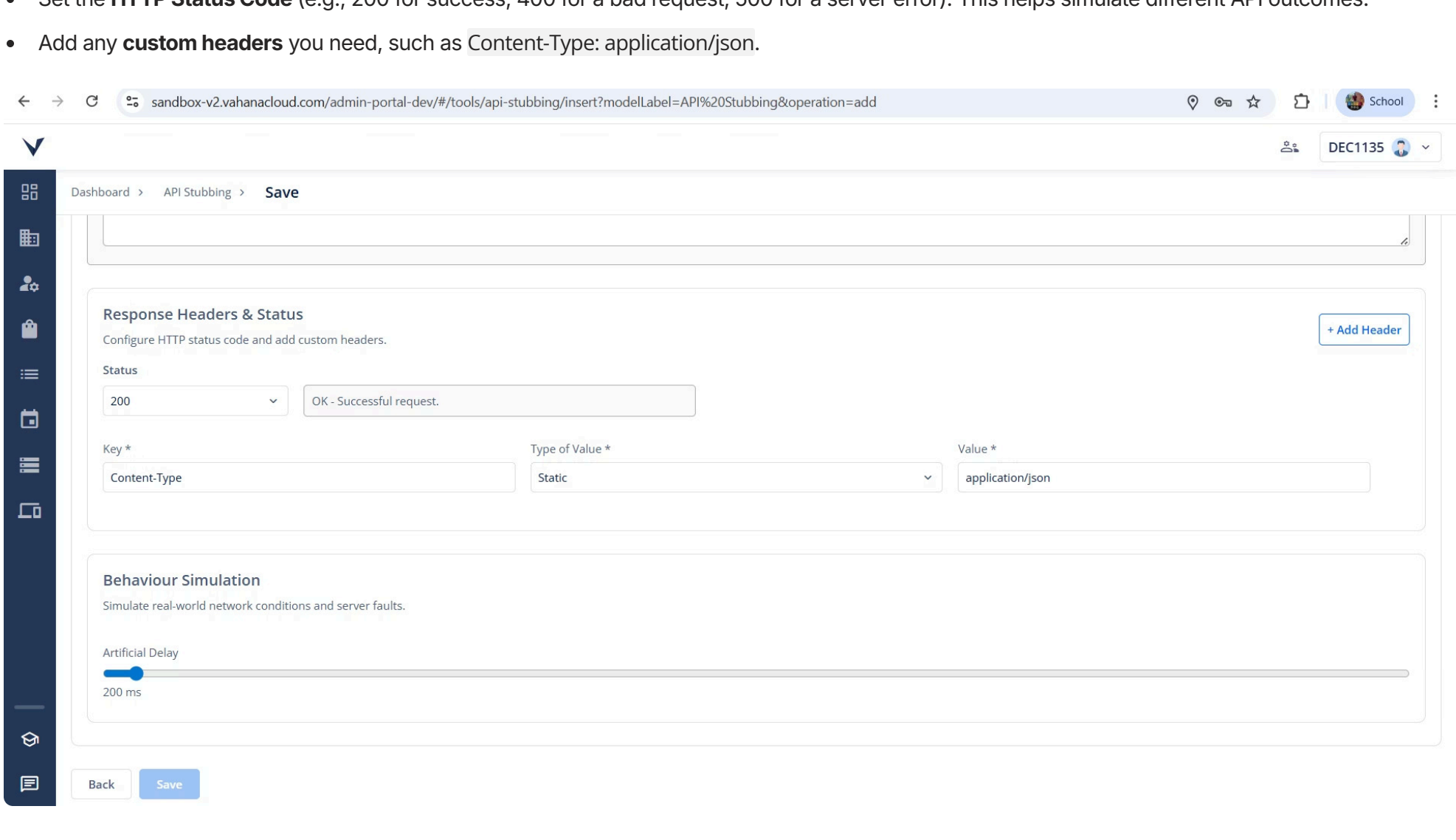

Fig 6: [Option to add the HTTP response status code and custom response headers along with artificial delay]

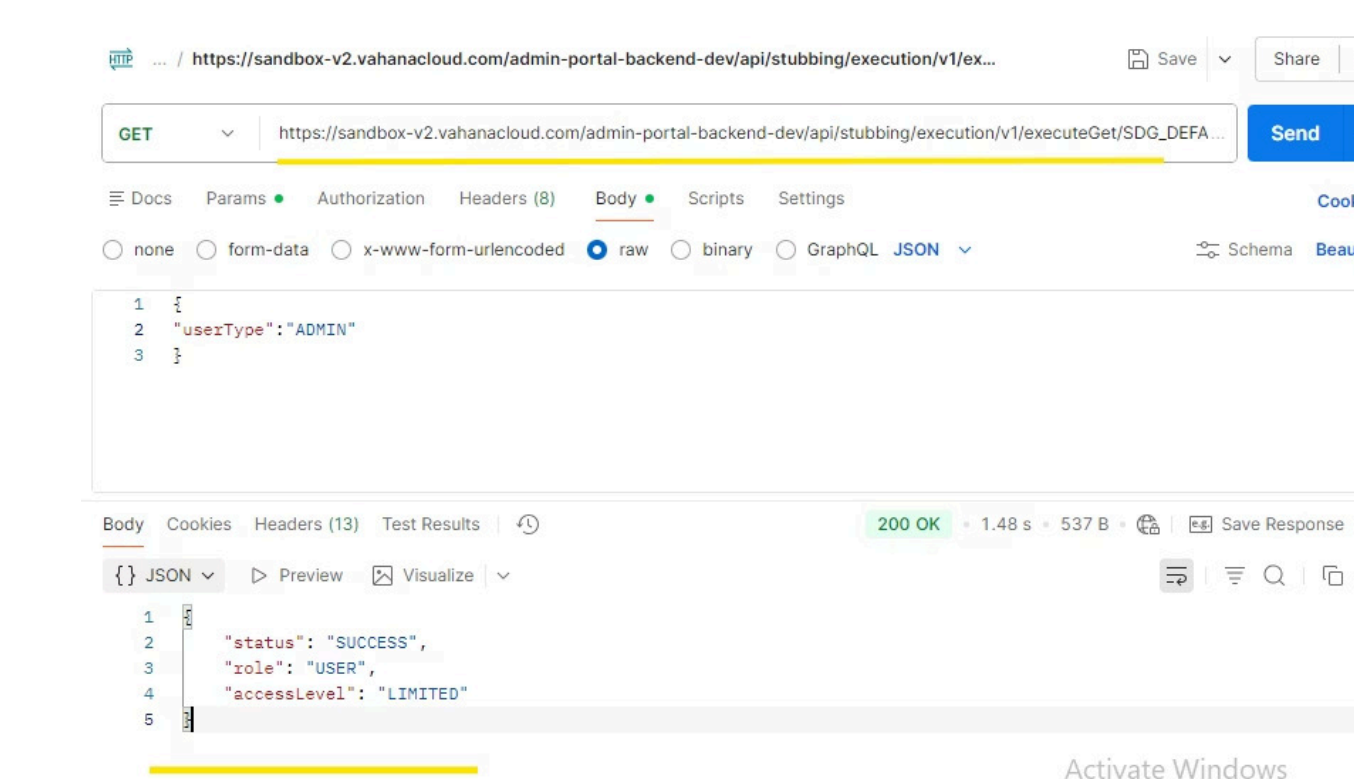

Fig 7: [Screenshot: Copy **URL Path Pattern** into **Postman** new request and check customize API responses]

Step 6: Behavior Simulation

Test how strong your application is when things go wrong:

- Add **Artificial Delay** to mimic slow network speeds or backend services. This helps you see how users will experience delays and how your app handles timeouts. Refer Fig. 6

1.5 Stub Management

Easily handle your API stubs from start to finish:

- **Export**: Download your stub settings. You can then use them to migrate the stubbings in other environments.
- **Import**: Upload stubs you've exported before. The system checks them to make sure they work correctly.
- **Backup & Restore**: Create full backups of all your stubs and bring them back if needed. You can even go back to older versions, which helps protect against losing data by accident.

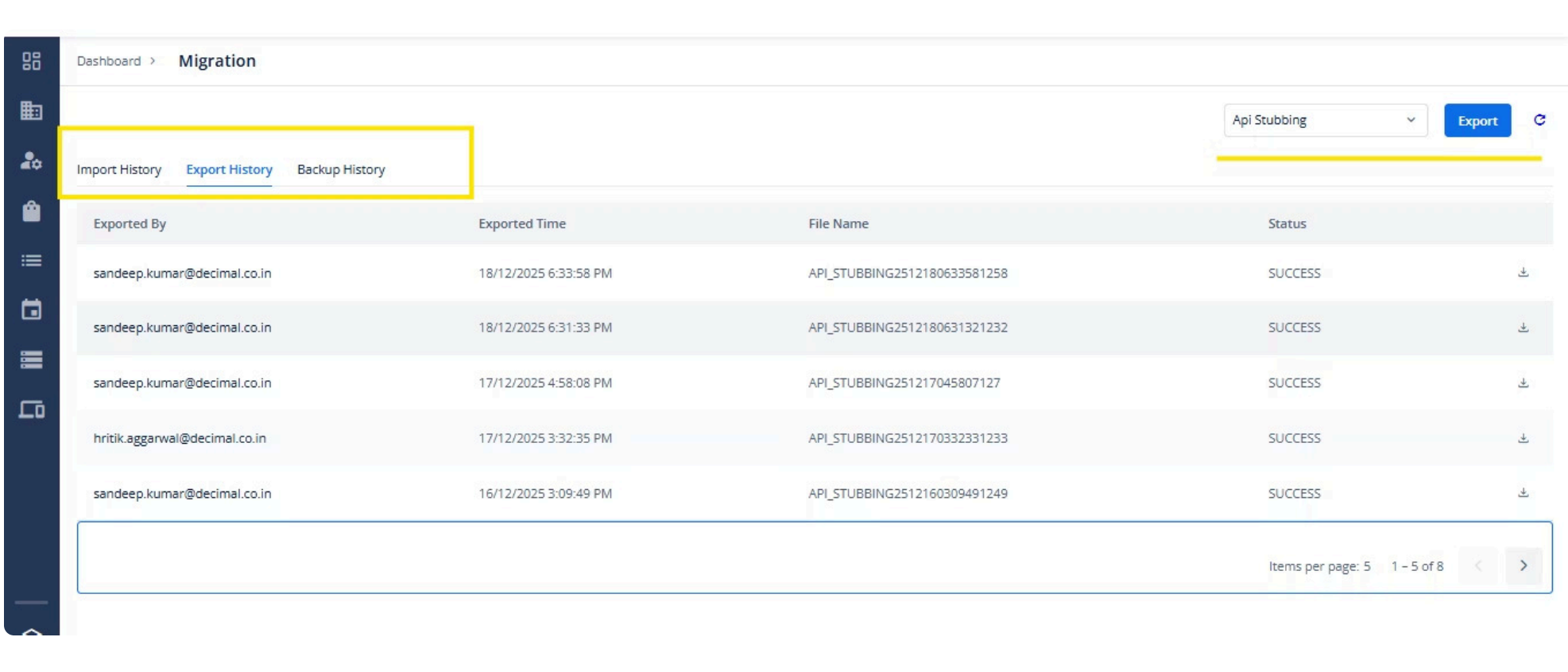

Fig 8 : [Screenshot: Export / Import / Backup. Highlights key tools for managing stubs]

1.6 Best Ways to Use API Stubbing

Get the most out of your API stubs with these tips:

- Use **Full Static Stubs** for simple tests where the response is always the same (e.g., success or basic error). This gives you a solid base for your tests.
- Choose **Conditional Stubs** for tests that involve different scenarios. For example, testing how your app works for different user types, with varied data, or specific request details.
- Use **Passthrough + Override** to mix real API responses with your own changes. This is good for testing unusual situations or simulating temporary backend problems without changing the actual API.
- Always double-check the **JSON Response Body** to make sure it's correct and follows the expected structure before saving your stub.
- Use **Behavior Simulation** (delay) to thoroughly test how tough your app is. Make sure it can handle slow networks and other real-world issues smoothly.
- API Stubbing allows users to test APIs after creation. In the **Test** feature, users can add headers and provide a JSON input request. Clicking **Test** shows the API response in JSON format.
- Who can update, create or delete the stubs using access control like all other modules of Admin portal. (Ref Fig. 10)

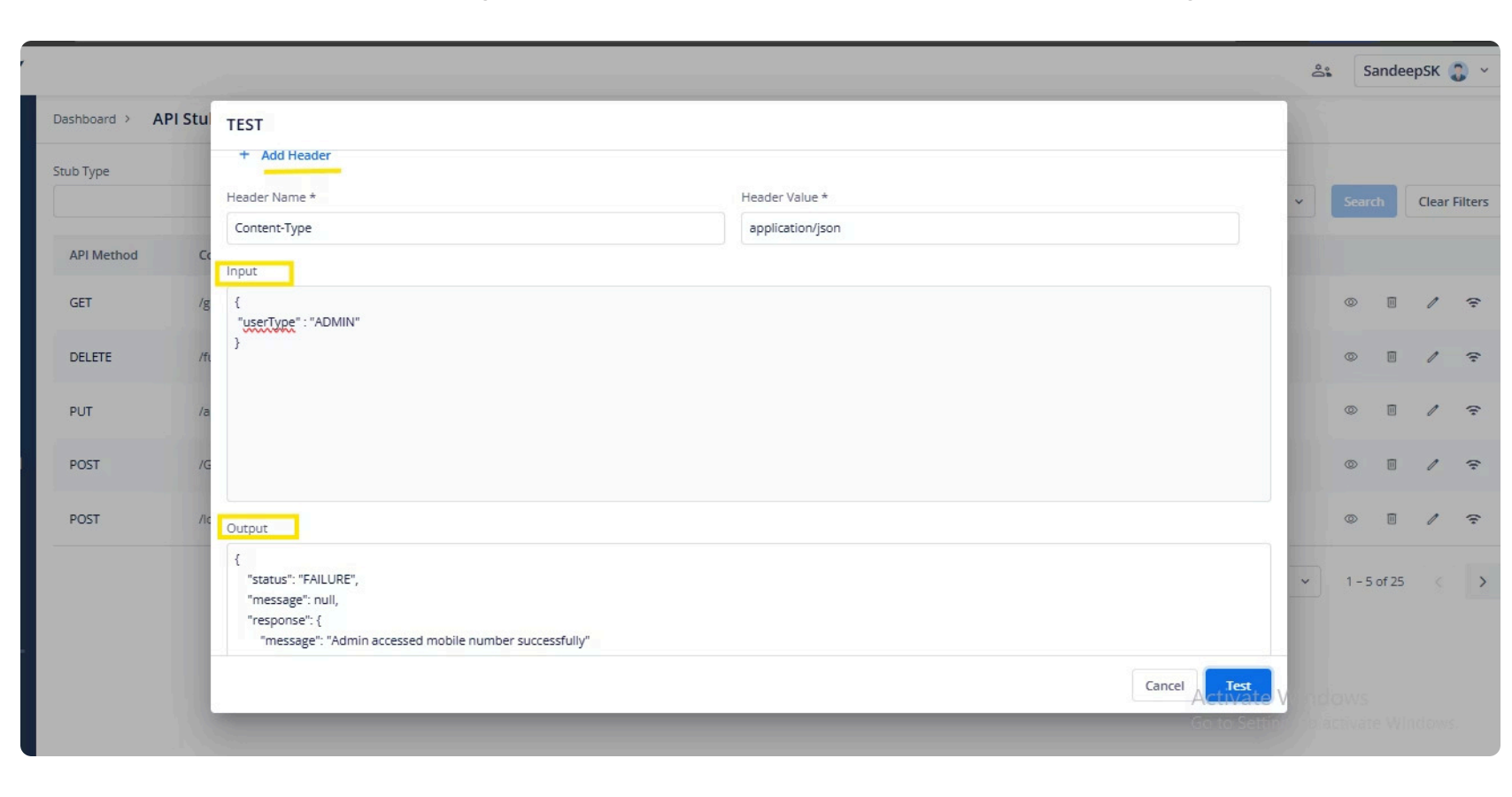

Fig 9 : [Screenshot Test your service: showing the add Header, Input payload, and Output response body]

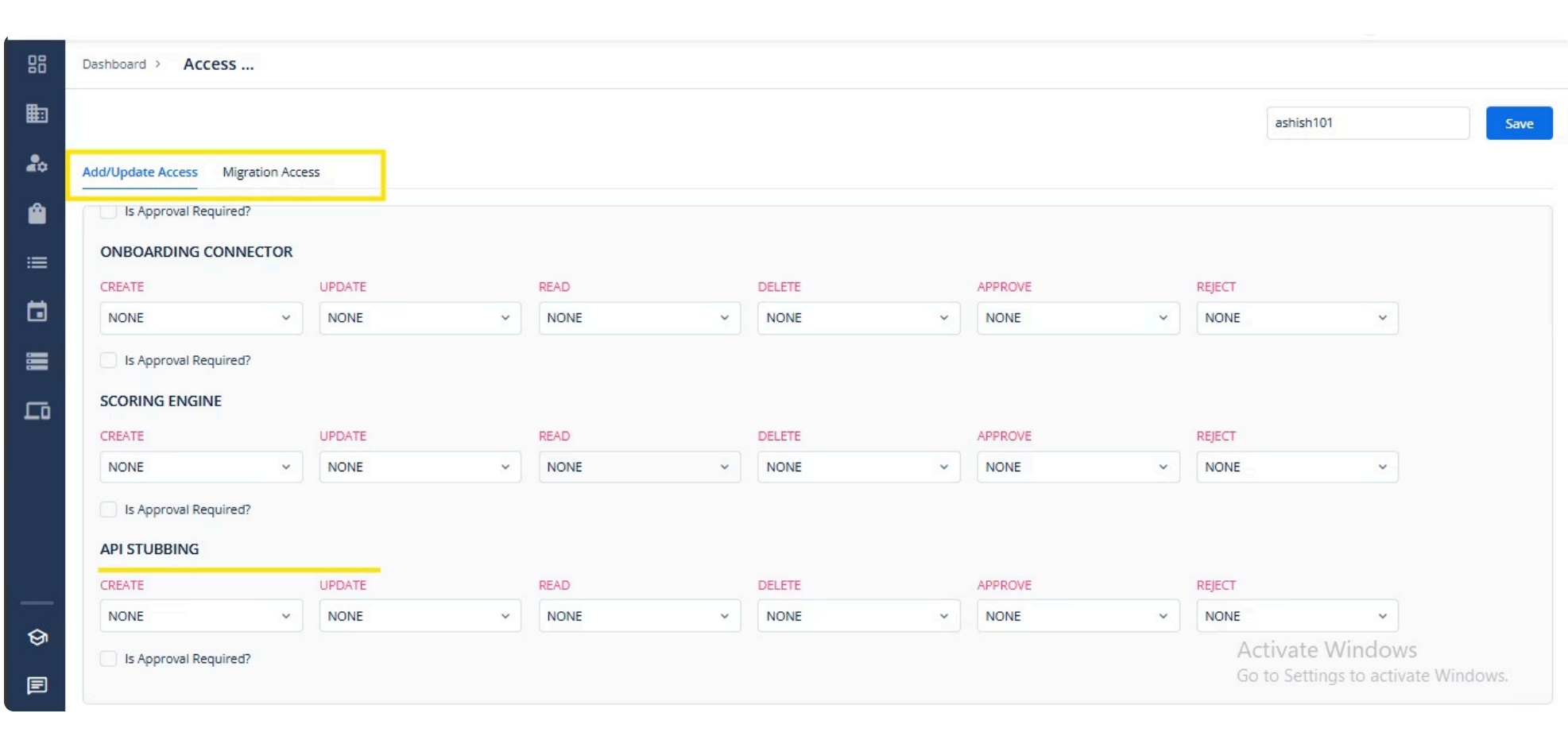

Fig 10 : [Screenshot Access Management. Shows how user permissions affect who can create and change stubs]

2. Database Performance Optimization (Enhancement)

This release also includes important backend improvements designed to make the system **faster, more stable, and better prepared for future growth**. These enhancements work behind the scenes and do not impact any user-facing functionality.

> ⓘ This enhancement was identified and raised by AU CASA Project team

2.1 Database Index Optimization

To improve system response time and handle growing data volumes more efficiently, database indexes have been optimized across key business tables.

Earlier, multiple separate indexes were used for individual fields. These have now been **combined into smarter, single composite indexes** that better reflect how the system actually searches and retrieves data. Unnecessary indexes have been safely removed from bfsi managed tables.

This approach reduces database effort while fetching data and improves overall performance. Below are the indexes of bfsi managed tables.

business_txn_data

Current Index

- **TxnDataIndex1** *(record_identifier1, task_code, task_status_code)*

Deprecated / Removed Indexes

- TxnDataIndex2 → *(task_code)*
- TxnDataIndex3 → *(task_status_code)*
- TxnDataIndex4 → *(record_identifier1)*

aofs

Current Index

- **AofsIndex1** *(record_identifier_1, record_identifier_2, record_identifier_3)*

Deprecated / Removed Indexes

- AofsIndex2 → *(record_identifier_1, record_identifier_2)*
- AofsIndex3 → *(record_identifier_1)*

leads

Current Index

- **LeadsIndex1** *(record_identifier_1, record_identifier_2, record_identifier_3)*

Deprecated / Removed Indexes

- LeadsIndex2 → *(record_identifier_1, record_identifier_2)*
- LeadsIndex3 → *(record_identifier_1)*

stakeholders

Current Index

- **ShIndex1** *(record_identifier_1, record_identifier_2, record_identifier_3)*

Deprecated / Removed Indexes

- ShIndex2 → *(record_identifier_1, record_identifier_2)*
- ShIndex3 → *(record_identifier_1)*

Overall Impact:

- Faster data retrieval for common business operations
- Developers can add custom index as per their requirements.

2.2 Database Connection Pool Optimization

To improve system reliability and prevent database connectivity issues, connection handling has been optimized.

New timeout settings ensure that unused connections are cleaned up automatically and long-running connections are refreshed in a controlled manner. This helps avoid unecessary closure of idle connections.

Added Configuration Parameters

- **max.idle.time.ms** Automatically closes unused database connections after X minutes to free resources. The value need to be enter in milliseconds.
- **max.life.time.ms** Refreshes long-running database connections to avoid stale or dropped connections after X minutes. The value need to be enter in milliseconds.

Overall Impact:

- Bring down connections to minimum when load is less.
- Reduced risk of unexpected connection failures

> ⓘ This issue was identified by AU CASA Team.

Quick Fixes and Common Questions

If you run into any problems with the new features, try these steps first:

- **Clear Your Browser Cache:** A full refresh of your browser (Ctrl+Shift+R or Cmd+Shift+R) can fix many display issues. It helps load the newest parts of the website instead of old cached versions.

Your Feedback Matters

We are committed to continuous improvement. Feedback on the new interface is highly valued and will drive future iterations.

- **Community Forum:** Post detailed suggestions and engage in discussion within the 'Vahana Community' tab.

Future updates will focus on making things faster, refining the design even more, and adding the features you ask for most.

Conclusion 📑

The improved API Stubbing module lets you easily create simulated API responses, helping you test your software faster and more reliably without relying on real services. It gives you full control over how your tests behave, from simple fixed responses to complex conditional scenarios and even simulating real-world network issues.

We encourage you to dive in and explore these powerful new tools. They're designed to make your development workflow smoother and your testing more robust!

Supported Build Versions

- **Workflow Manager**: 17.0.0-1.0.218
- **Admin Portal (Front End)**: 18.0.0-1.0.255-production
- **Admin Portal (Back End)**: 17.0.0-1.0.287
- **BFSI**: 17.0.0-1.0.211

Need Help?

For any queries or support related to this release, please contact the "**Admin Portal**" team.