Test Document

for

PHInance

Version 1.00

Prepared by

Group 15: **Group Name:** Team FEINance

Anisha Srivastava	220149	anishas22@iitk.ac.in
Dilbar Singh Lamba	220368	dilbars22@iitk.ac.in
Pranshu Thirani	220801	pranshut22@iitk.ac.in
Sameer Yadav	220950	sameer22@iitk.ac.in
Sangam Gupta	220961	sangamg22@iitk.ac.in
Aruz Awasthi	230209	aruza23@iitk.ac.in
Anirudh	220146	anirudhm22@iitk.ac.in
Aadi Singh	220004	aadis22@iitk.ac.in
Aadya Dhir	220010	aadyad22@iitk.ac.in
Anish Sahu	220148	anishs22@iitk.ac.in

Course: CS253

Mentor TA: Naman Baranwal

Date: 05/04/2025

Content

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.00	Anisha Srivastava Dilbar Singh Lamba Pranshu Thirani Sameer Yadav Sangam Gupta Aruz Awasthi Anirudh Aadi Singh Aadya Dhir Anish Sahu	The First Version of the Test Document	05/04/2025

1. Introduction

Test Strategy: We used manual testing to test our software.
Testing period: The majority of the testing was done after the implementation was completed. However, even during the implementation phase, we frequently did some manual testing of our code.
The Testers: Developers were themselves the testers. However, we ensured that the person who wrote a particular functionality has not tested the same functionality.
Coverage Criteria: We have used functional and non-functional coverage.

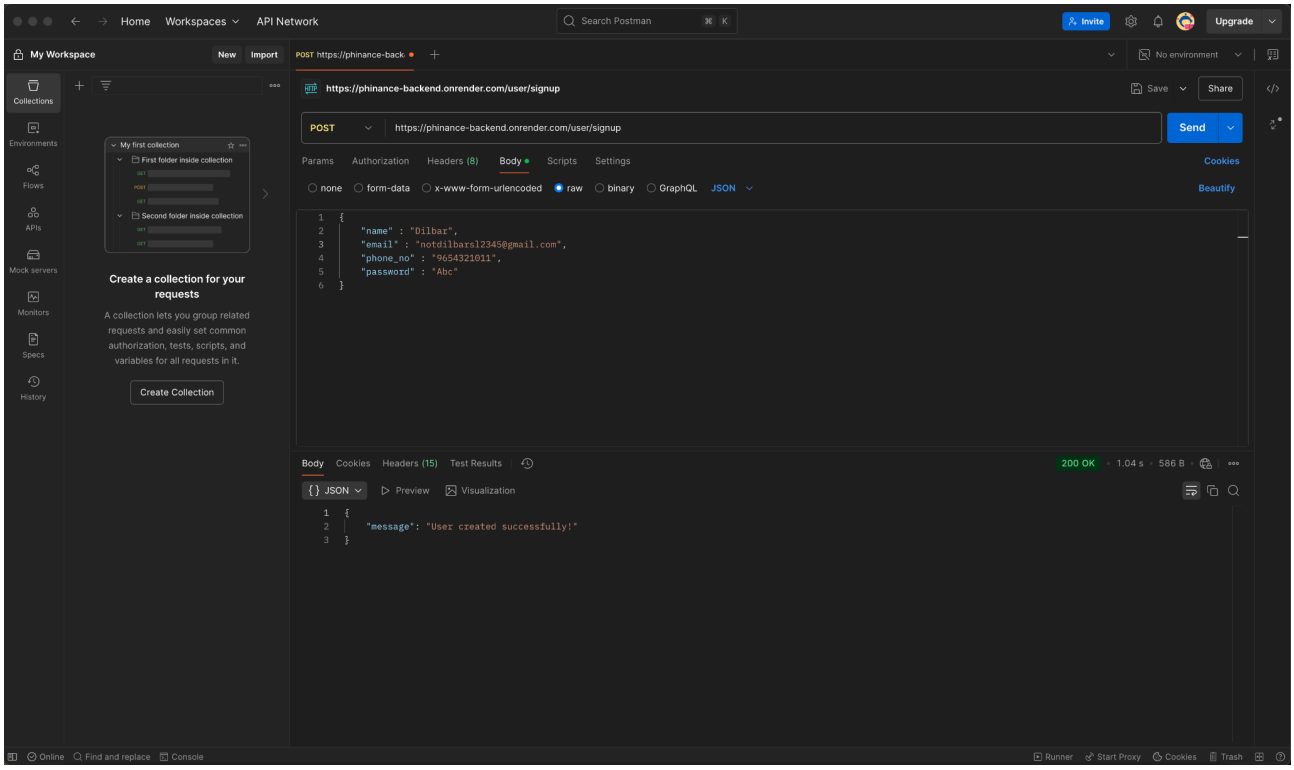
Tools used for testing:

We used **Postman** to test all our APIs and backend functionality.
Postman is a powerful tool that makes it really easy to send requests and check responses without writing any code.It's key advantages include:

- **Easy to Use:** Postman has a simple and clean interface, which made it easy for us to test different APIs quickly without any setup hassle.
- **Great for Testing:** It lets you write small scripts to test different conditions, pass data between requests, and even automate checks.
- **Good for Teams:** Postman makes it easy to share collections and test cases with the team, so we're all on the same page.
- **Keeps Things Organized:** We could keep all our requests, responses, and environments in one place—great for going back and retesting things later.
- **Free & Available Everywhere:** It's free to use and works across all major platforms, so it fits right into our workflow.

Using **Postman** made our testing smoother and more efficient, especially for trying out different user scenarios and checking how the backend handles them.

2. Unit and Integration Testing



Unit Name: User Signup API Endpoint Test

Basic Information: Tested the /user/signup POST endpoint of the phinance-backend.onrender.com server using Postman. A new user was created successfully using a JSON payload.

Unit Details:

Class/Function: user_signup function in the backend API

Endpoint: POST /user/signup

Payload:

```
{
 "name": "Dilbar",
 "email": "notdilbars12345@gmail.com",
 "phone_no": "9654321011",
 "password": "Abc"
}
```

Test Owner : Anirudh

Test Date : 01/04/2025

Test Results:

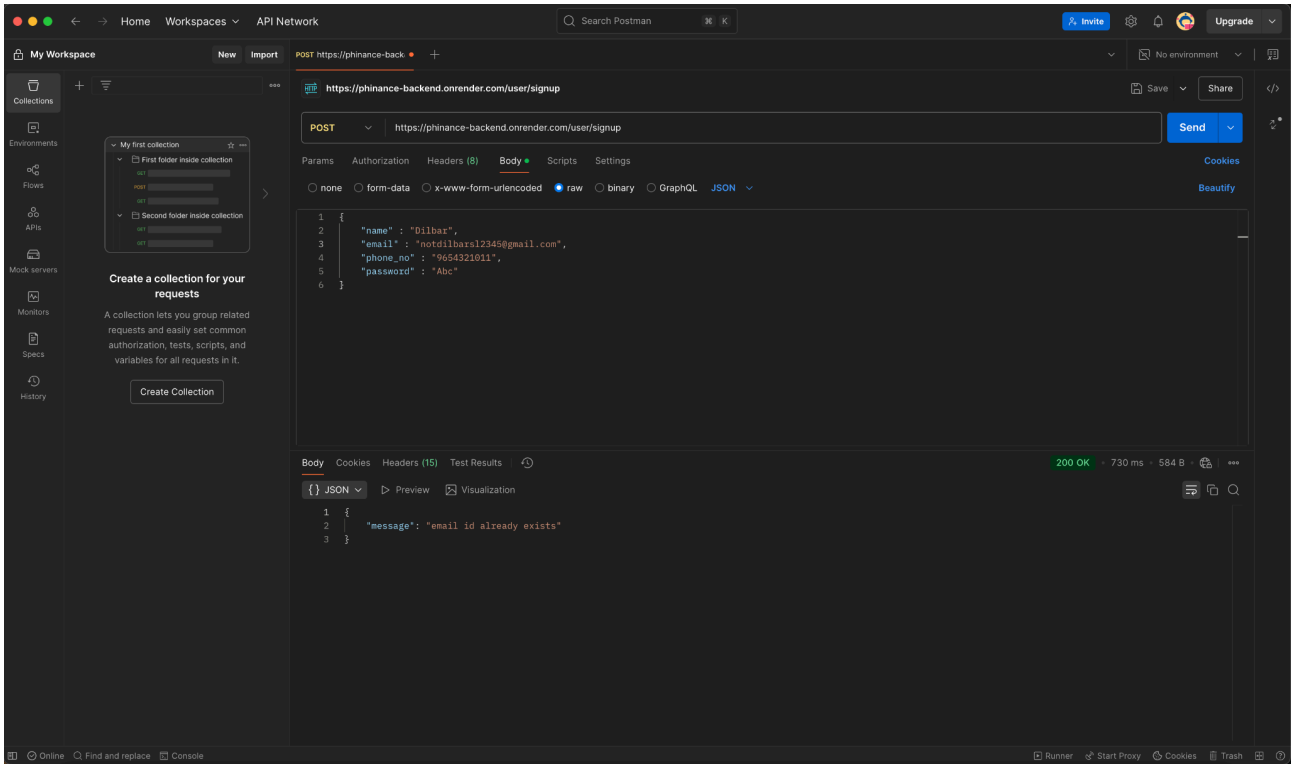
The test was successful. The API returned a 200 OK status with the response:

{
"message": "User created successfully!"
}

This confirms the user registration logic is functioning as expected with valid input data.

Structural Coverage

- Positive path tested with valid input.
- Request format: JSON (raw)
- HTTP status and response body verified.
- No validation or failure cases tested in this instance.



Unit Name: User Signup API Endpoint Test - Duplicate Email Case

Basic Information: Tested the /user/signup POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to create a user with an email that already exists in the system.

Unit Details:

Class/Function: user_signup function in the backend API

Endpoint: POST /user/signup

Test Owner: Aadya Dhir

Test Date: 02/04/2025

Test Results

The test returned a 200 OK status with the following response:

{

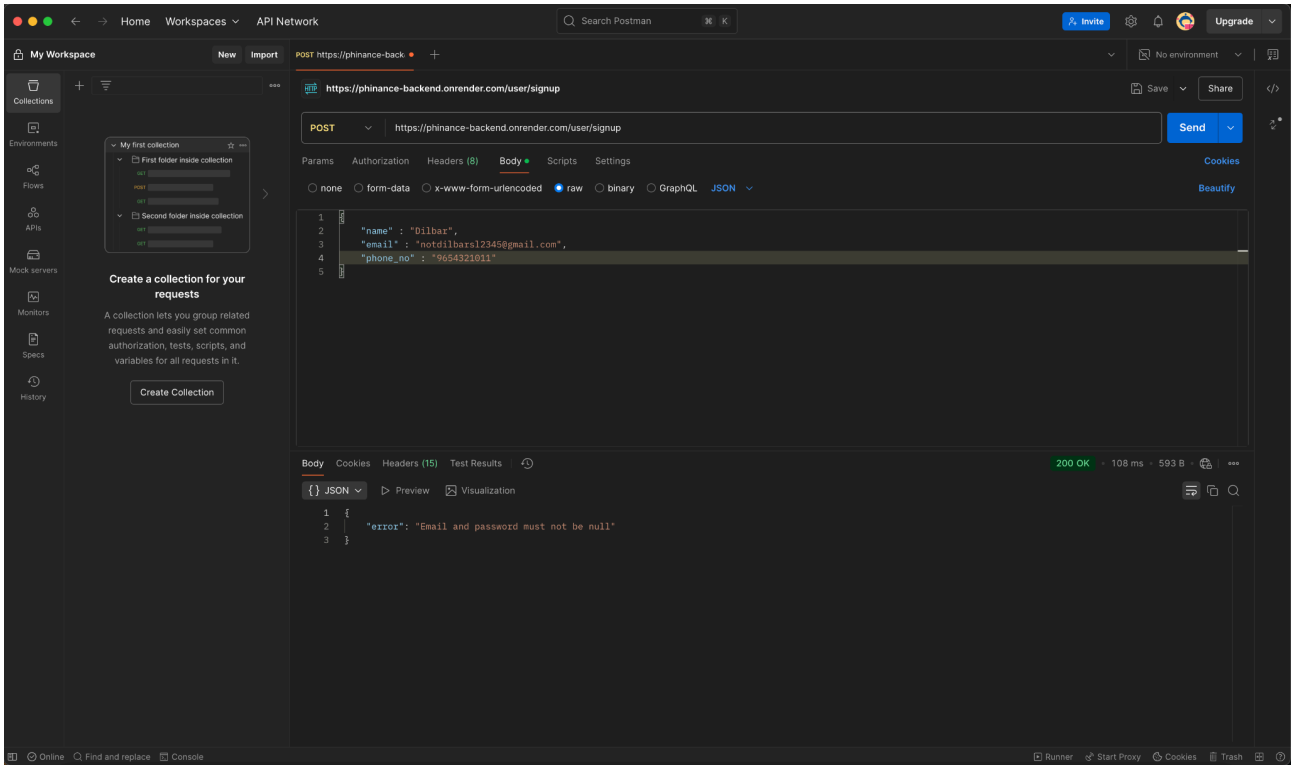
"message": "email id already exists"

}

This confirms that the API correctly identifies duplicate email addresses and prevents re-registration using the same email.

Structural Coverage

- Negative path tested with duplicate input.
- Request format: JSON (raw)
- HTTP status and validation response body verified.
- Validates backend logic for user uniqueness.



Unit Name: User Signup API Endpoint Test - Missing Fields Case

Basic Information: Tested the /user/signup POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to create a user without providing a password field in the JSON payload.

Unit Details:

Class/Function: user_signup function in the backend API

Endpoint: POST /user/signup

Payload:

{

"name": "Dilbar",
"email": "notdilbars12345@gmail.com",
"phone_no": "9654321011"
}

Test Owner: Aruz Awasthi

Test Date: 04/04/2025

Test Results:

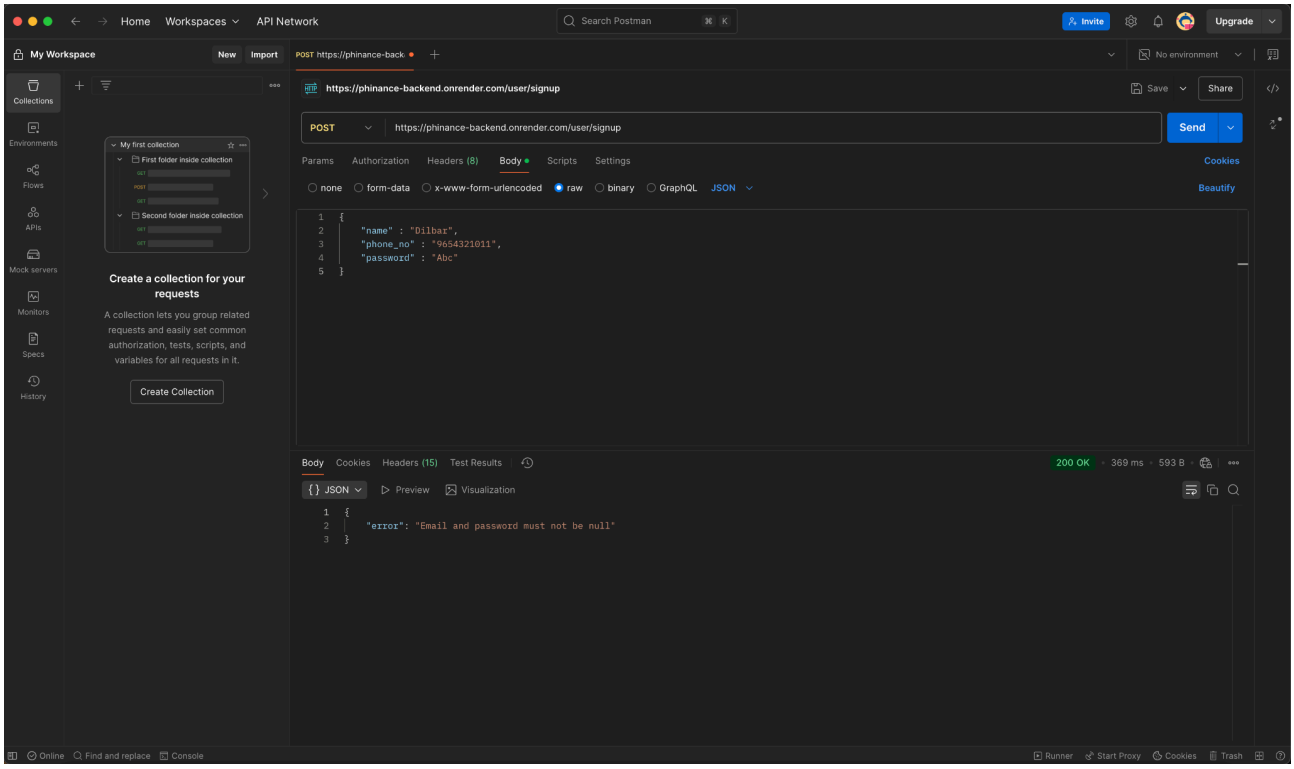
The test returned a 200 OK status with the following response:

{
"error": "Email and password must not be null"
}

This confirms that the API correctly identifies missing required fields and prevents user creation when password or email is null.

Structural Coverage

- Negative path tested with missing required input.
- Request format: JSON (raw)
- HTTP status and error message validated.
- Confirms server-side validation logic is present.



Unit Name: User Signup API Endpoint Test - Missing Email Case

Basic Information: Tested the /user/signup POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to create a user without providing an email field in the JSON payload.

Unit Details:

Class/Function: user_signup function in the backend API

Endpoint: POST /user/signup

Payload:

```
{
  "name": "Dilbar",
  "phone_no": "9654321011",
  "password": "Abc"
}
```

Test Owner: Pranshu Thirani

Test Date: 02/04/2025

Test Results:

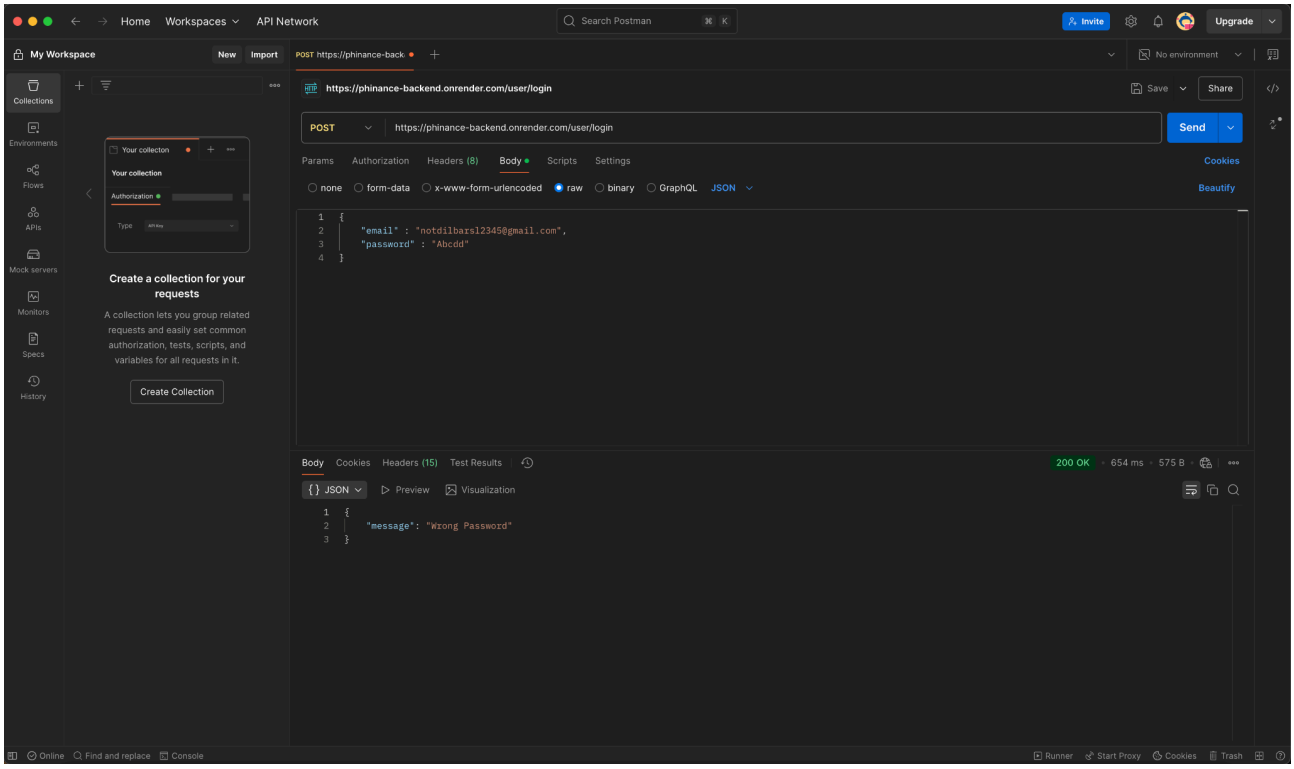
The test returned a 200 OK status with the following response:

{
"error": "Email and password must not be null"
}

This confirms that the API correctly checks for the presence of required fields and blocks requests that omit the email.

Structural Coverage:

- Negative path tested with missing 'email' field.
- Request format: JSON (raw)
- HTTP status and error message verified.
- Validates server-side validation logic for required user input.



Unit Name: User Login API Endpoint Test - Wrong Password Case

Basic Information: Tested the /user/login POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to log in using a valid email with an incorrect password.

Unit Details:

Class/Function: user_login function in the backend API

Endpoint: POST /user/login

Payload:

```
{
  "email": "notdilbars12345@gmail.com",
  "password": "Abcdd"
}
```

Test Owner:Dilbar Singh Lamba

Test Date:31/03/2025

Test Results:

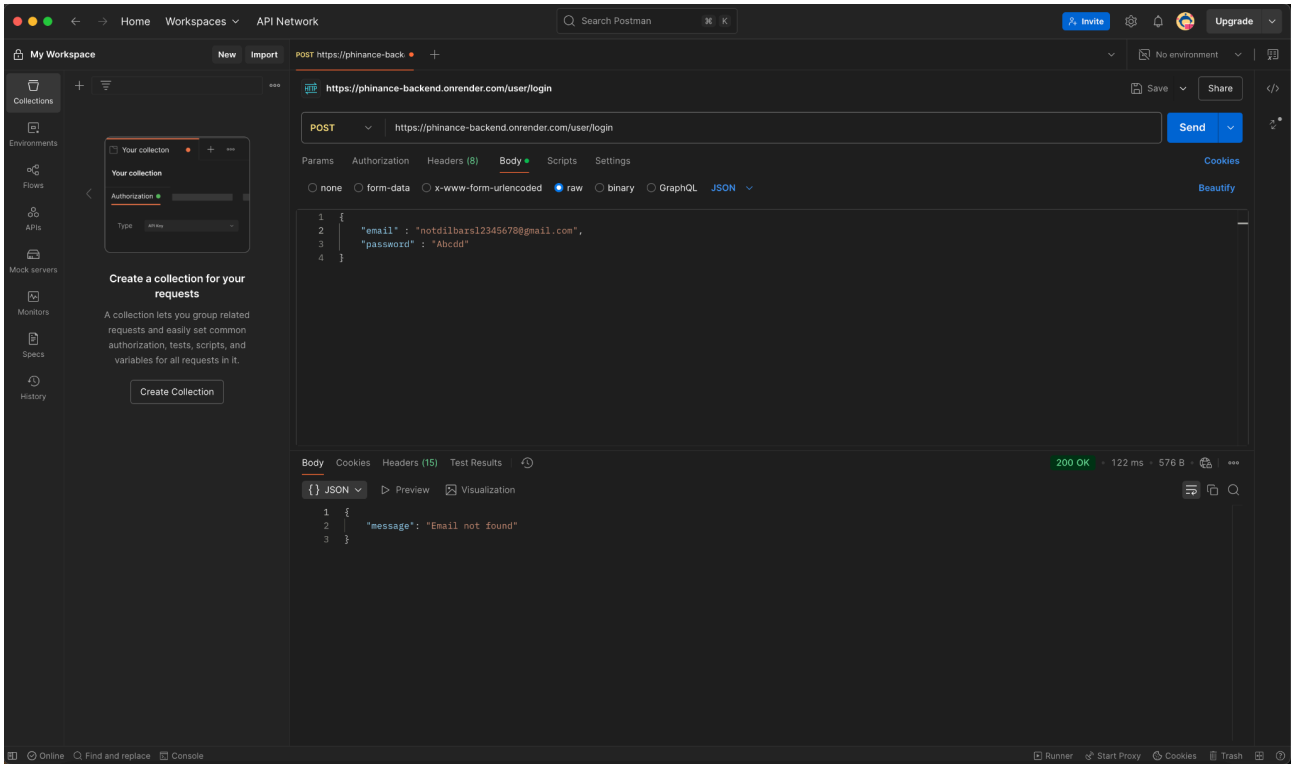
The test returned a 200 OK status with the following response:

{
"message": "Wrong Password"
}

This confirms that the API correctly validates login credentials and rejects access when the password is incorrect.

Structural Coverage:

- Negative path tested with incorrect authentication credentials.
- Request format: JSON (raw)
- HTTP status and response message validated.
- Confirms proper credential checking and error messaging.



Unit Name: User Login API Endpoint Test - Email Not Found Case

Basic Information: Tested the /user/login POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to log in with an email address that does not exist in the system.

Unit Details:

Class/Function: user_login function in the backend API

Endpoint: POST /user/login

Payload:

```
{
  "email": "notdilbars12345678@gmail.com",
  "password": "Abcdd"
}
```

Test Owner: Anish Sahu

Test Date: 02/04/2025

Test Results

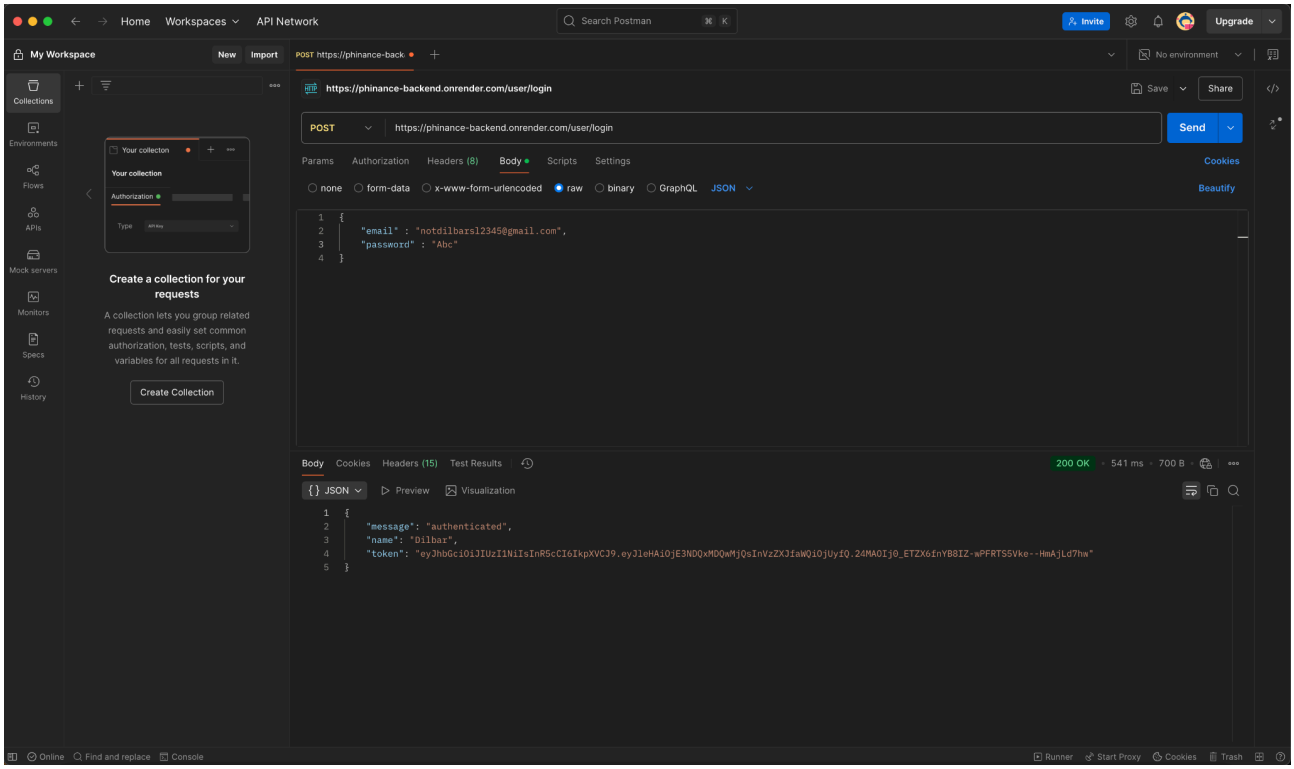
The test returned a 200 OK status with the following response:

{
"message": "Email not found"
}

This confirms the API correctly checks whether the email exists and prevents login for unregistered users.

Structural Coverage

- Negative path tested with a non-existent email.
- Request format: JSON (raw)
- HTTP status and message validated.
- Validates user lookup and proper feedback for missing accounts.



Unit Name

User Login API Endpoint Test - Successful Login Case

Basic Information: Tested the /user/login POST endpoint of the phinance-backend.onrender.com server using Postman. Successfully logged in using valid credentials.

Unit Details

Class/Function: user_login function in the backend API

Endpoint: POST /user/login

Payload:

```
{
  "email": "notdilbars12345@gmail.com",
  "password": "Abc"
}
```

Test Owner: Sameer Yadav

Test Date: 01/04/2025

Test Results

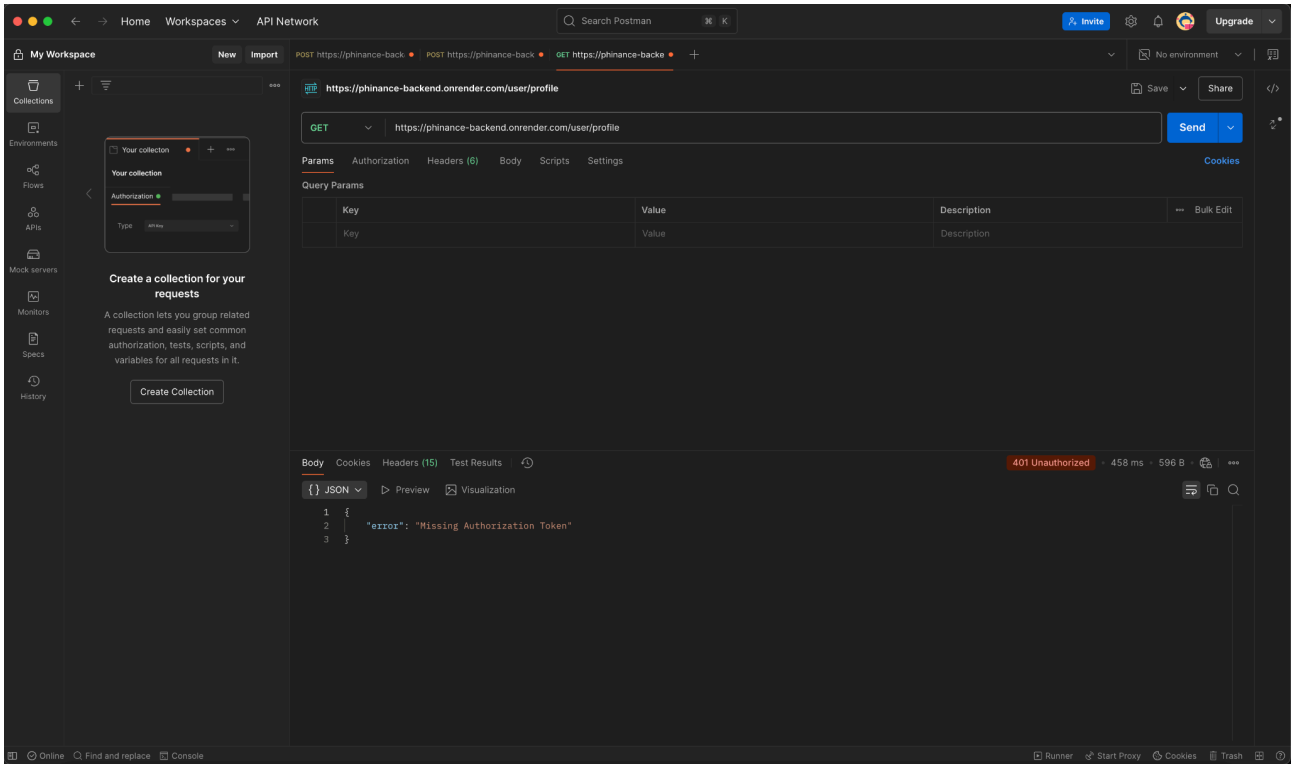
The test returned a 200 OK status with the following response:

```
{
  "message": "authenticated",
  "name": "Dilbar",
  "token": "<JWT token>"
}
```

This confirms that the login API correctly authenticates a user with valid credentials and returns a JSON Web Token (JWT) for session management.

Structural Coverage

- Positive path tested with valid login input.
- Request format: JSON (raw)
- HTTP status, user details, and token generation verified.
- Demonstrates full flow of authentication and session creation.



Unit Name: User Profile API Endpoint Test - Missing Authorization Token

Basic Information: Tested the /user/profile GET endpoint of the phinance-backend.onrender.com server using Postman. Attempted to access user profile data without providing an Authorization token.

Unit Details:

Class/Function: get_user_profile function in the backend API

Endpoint: GET /user/profile

Test Owner: Anisha Srivastava

Test Date: 31/03/2025

Test Results:

The test returned a 401 Unauthorized status with the following response:

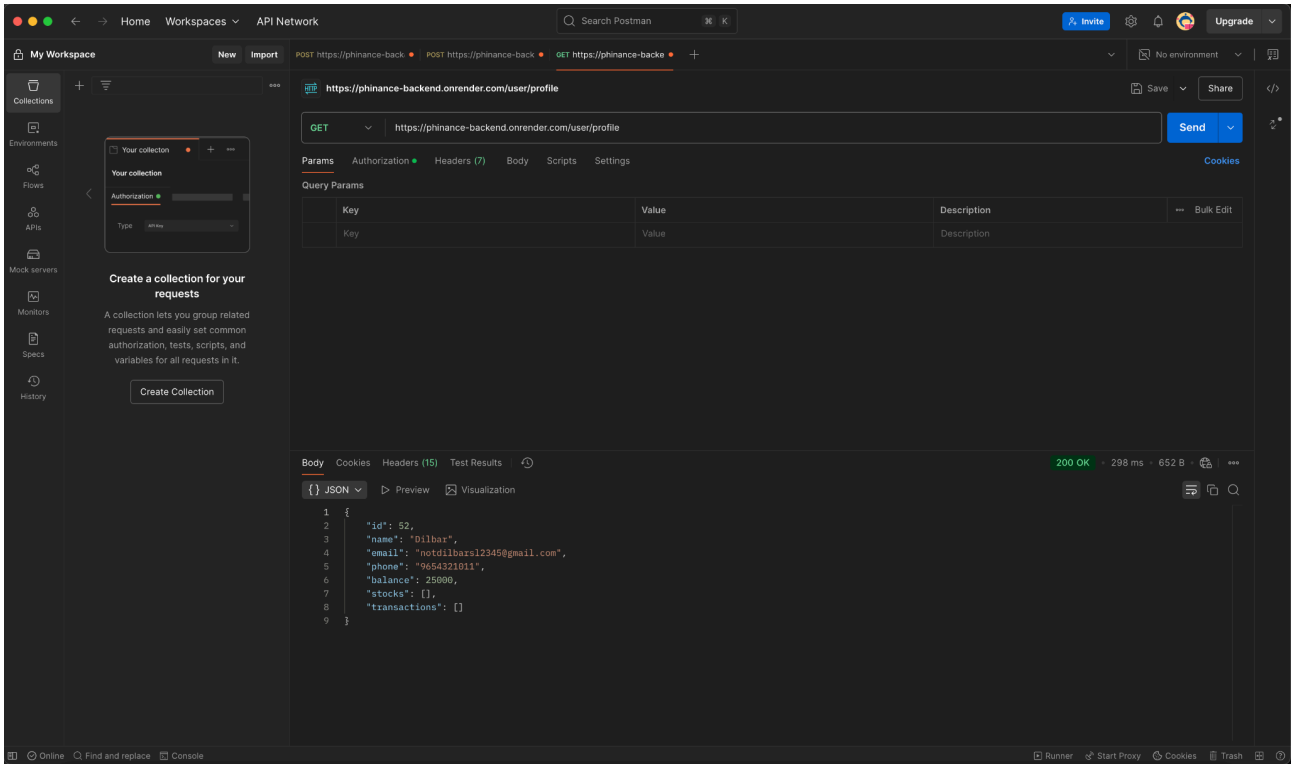
```
{
  "error": "Missing Authorization Token"
```

}

This confirms that the API enforces authentication requirements and prevents unauthorized access to protected endpoints.

Structural Coverage

- Negative path tested without an Authorization header.
- HTTP status and error response validated.
- Confirms security check is in place for token-based access control.



Unit Name: User Profile API Endpoint Test - Successful Access Case

Basic Information: Tested the /user/profile GET endpoint of the phinance-backend.onrender.com server using Postman. Successfully accessed user profile data using a valid Authorization token.

Unit Details:

Class/Function: get_user_profile function in the backend API

Endpoint: GET /user/profile

Test Owner: Anirudh

Test Date: 03/04/2025

Test Results:

The test returned a 200 OK status with the following response:

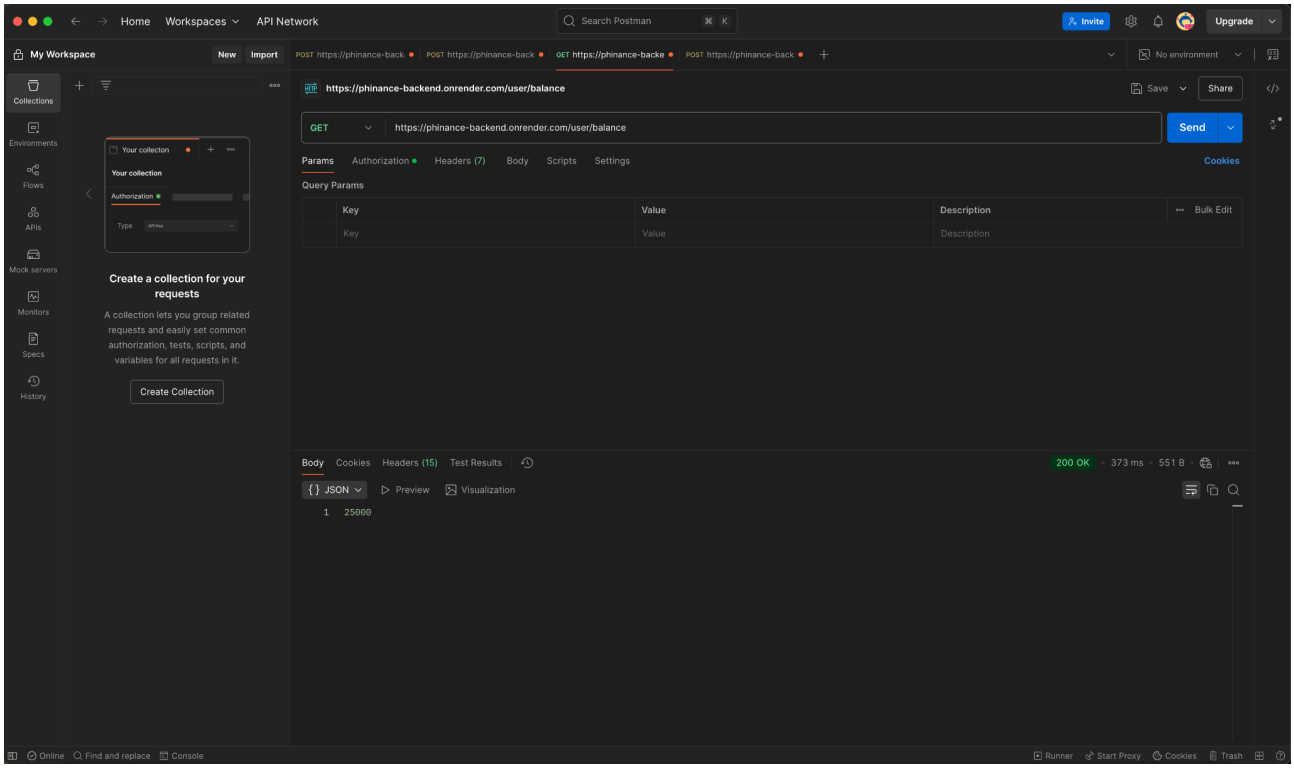
{
"id": 52,
"name": "Dilbar",

"email": "notdilbars12345@gmail.com",
"phone": "9654321011",
"balance": 25000,
"stocks": [],
"transactions": []
}

This confirms the profile endpoint correctly authorizes requests and returns user data securely when a valid token is supplied.

Structural Coverage

- Positive path tested with valid authorization.
- Authorization header, token verification, and data retrieval validated.
- End-to-end security and access flow confirmed for authenticated users.



Unit Name: User Balance API Endpoint Test - Successful Access Case

Basic Information: Tested the /user/balance GET endpoint of the phinance-backend.onrender.com server using Postman. Successfully retrieved user balance using a valid Authorization token.

Unit Details:

Class/Function: get_user_balance function in the backend API

Endpoint: GET /user/balance

Test Owner: Sangam Gupta

Test Date: 04/04/2025

Test Results:

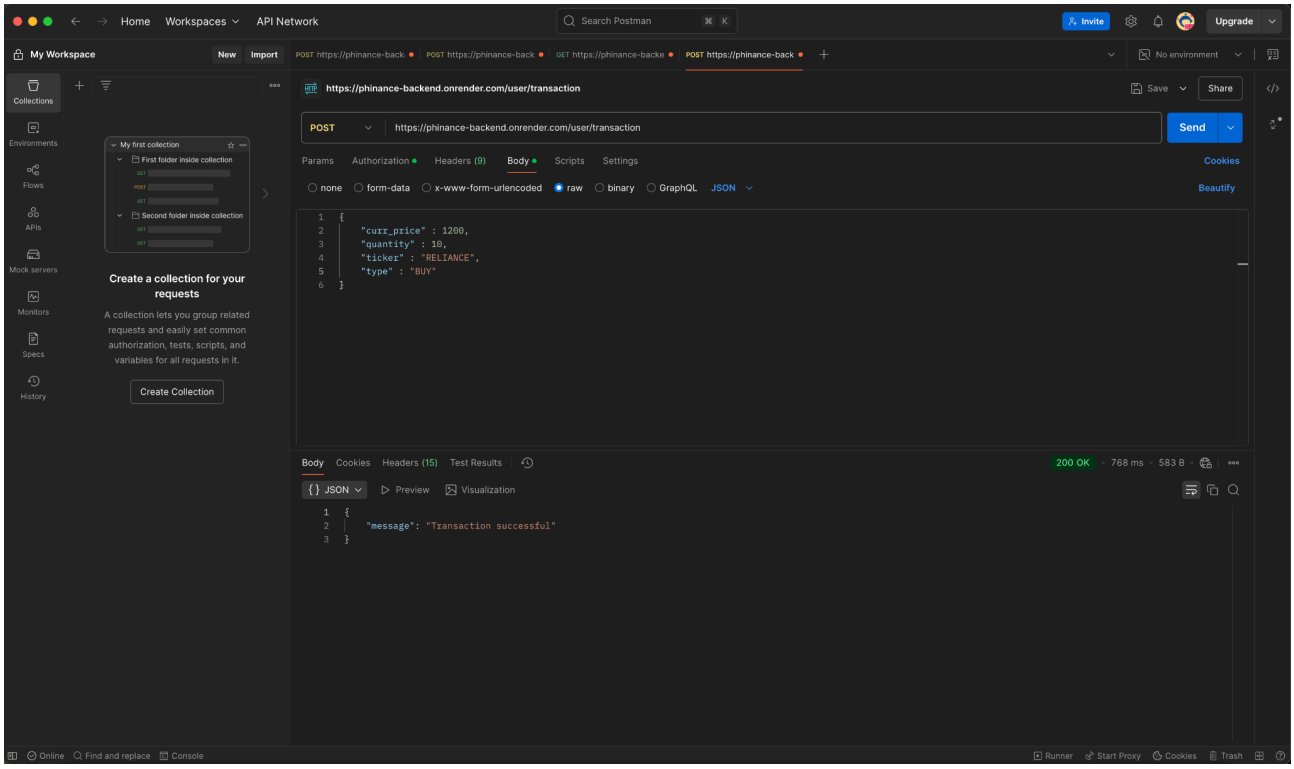
The test returned a 200 OK status with the following response:

25000

This confirms the balance endpoint correctly authenticates the request and returns the user's current balance when a valid token is supplied.

Structural Coverage

- Positive path tested with valid token.
- Authorization header, token validation, and numeric response confirmed.
- Demonstrates secure access to financial data.



Unit Name: User Transaction API Endpoint Test - Successful Buy Order

Basic Information: Tested the /user/transaction POST endpoint of the phinance-backend.onrender.com server using Postman. Successfully executed a stock buy transaction.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

```
{
  "curr_price": 1200,
  "quantity": 10,
  "ticker": "RELIANCE",
  "type": "BUY"
}
```

Test Owner: Anish Sahu

Test Date: 04/04/2025

Test Results:

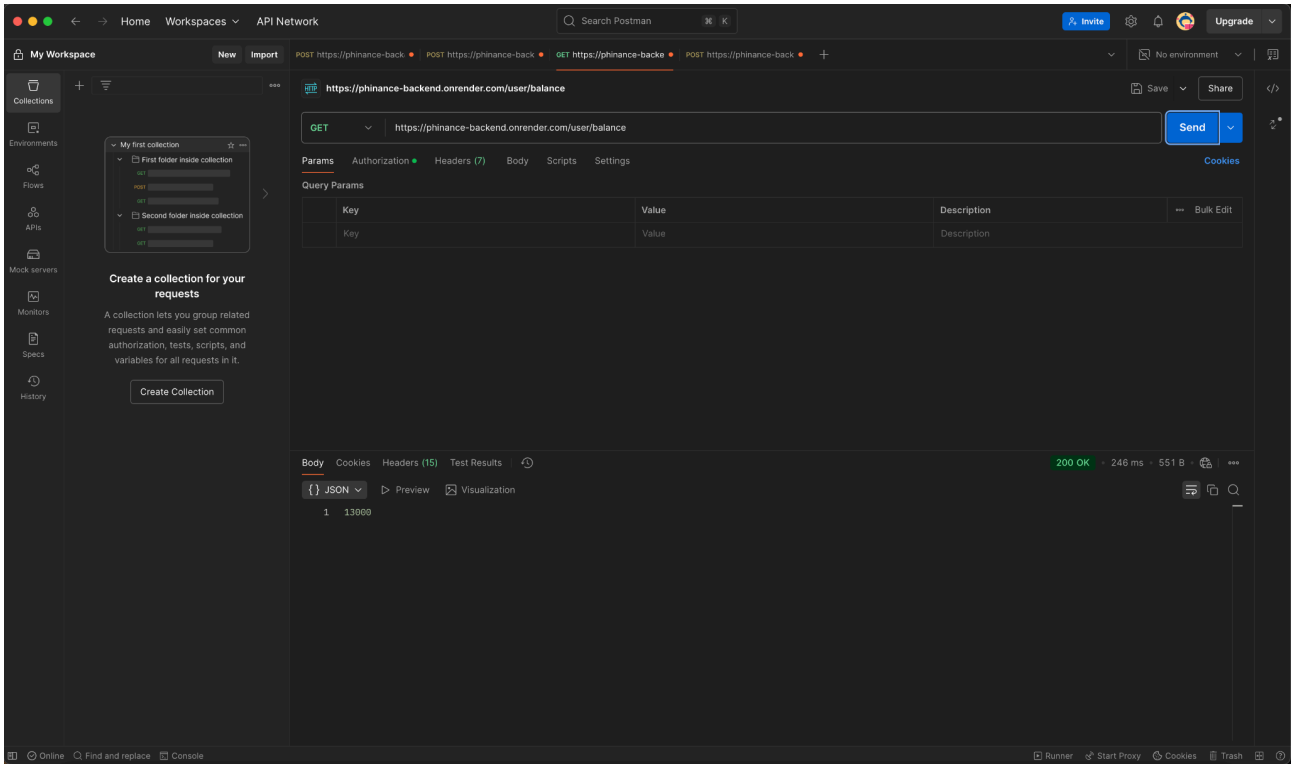
The test returned a 200 OK status with the following response:

{
  "message": "Transaction successful"
}

This confirms that the transaction endpoint processes and records valid buy orders as expected.

Structural Coverage:

- Positive path tested for valid stock purchase.
- Order creation, balance adjustment, and data persistence assumed validated through API response.
- Demonstrates full transaction lifecycle for a BUY request.



Unit Name: User Balance API Endpoint Test - Post-Transaction Balance Check

Basic Information: Tested the /user/balance GET endpoint of the phinance-backend.onrender.com server using Postman. Retrieved updated balance after executing a stock purchase transaction.

Unit Details:

Class/Function: get_user_balance function in the backend API

Endpoint: GET /user/balance

Test Owner:Aadya Dhir

Test Date: 03/04/2025

Test Results:

The test returned a 200 OK status with the following response:

13000

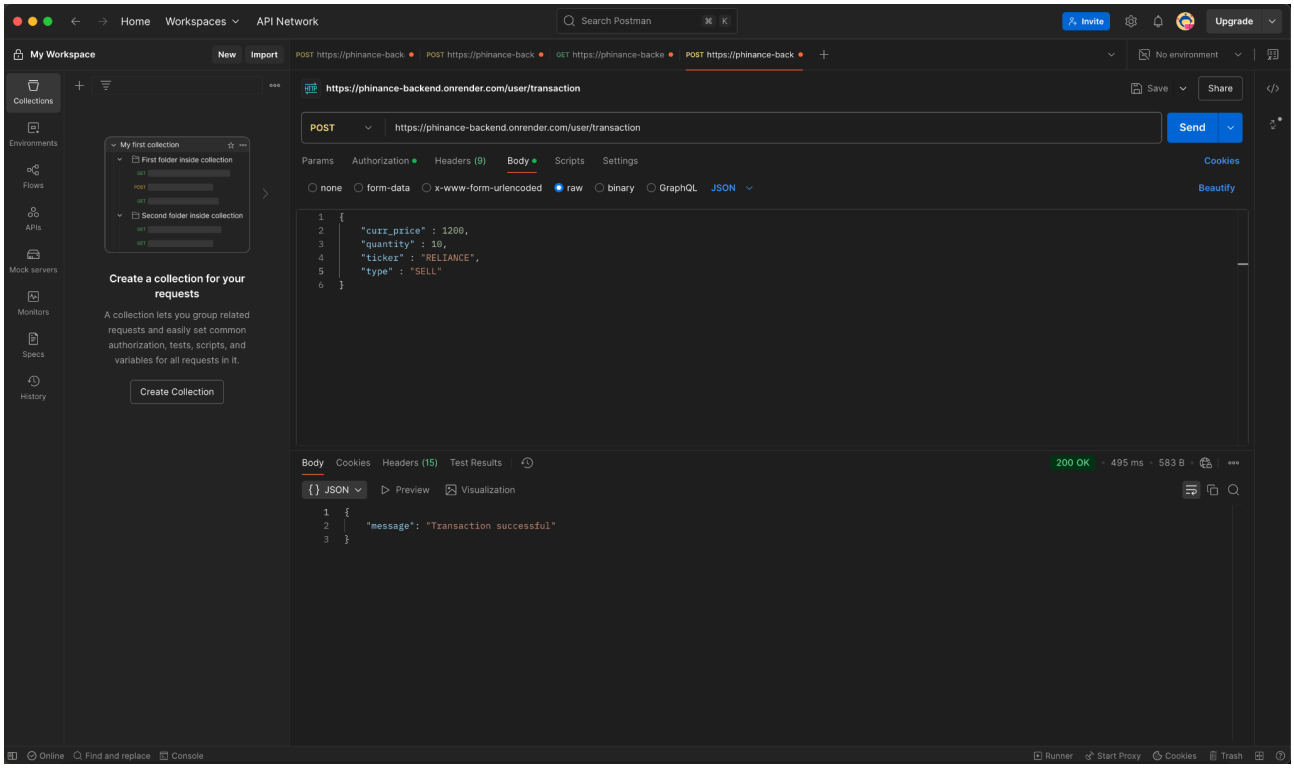
This confirms that the user's balance was accurately updated after a successful stock purchase (original balance was 25000; deducted 12000 for 10 shares at 1200 each).

Structural Coverage

- Post-action validation tested.

- Request authentication, balance deduction, and state persistence verified through subsequent balance read.

- Demonstrates correctness of financial transaction updates.



Unit Name: User Transaction API Endpoint Test - Successful Sell Order

Basic Information: Tested the /user/transaction POST endpoint of the phinance-backend.onrender.com server using Postman. Successfully executed a stock sell transaction.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

```
{
  "curr_price": 1200,
  "quantity": 10,
  "ticker": "RELIANCE",
  "type": "SELL"
}
```

Test Owner: Aadi Singh

Test Date: 02/04/2025

Test Results:

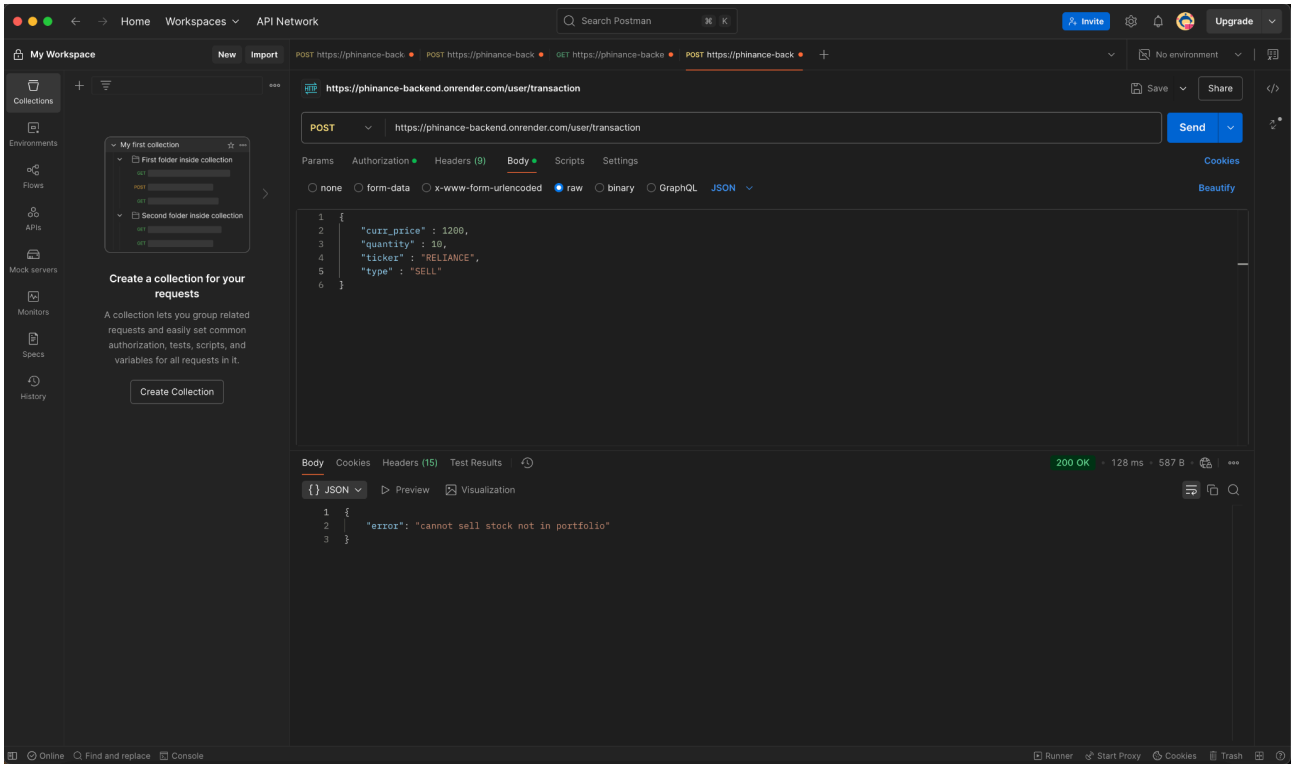
The test returned a 200 OK status with the following response:

{
"message": "Transaction successful"
}

This confirms that the transaction endpoint processes and records valid sell orders correctly and responds with success.

Structural Coverage:

- Positive path tested for valid stock sale.
- Validates ownership logic, stock quantity handling, and balance credit.
- Full sell transaction lifecycle verified via API response.



Unit Name: User Transaction API Endpoint Test - Sell Stock Not in Portfolio

Basic Information: Tested the /user/transaction POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted to sell a stock that was not present in the user's portfolio.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

```
{
  "curr_price": 1200,
  "quantity": 10,
  "ticker": "RELIANCE",
  "type": "SELL"
}
```

Test Owner: Anirudh

Test Date: 02/04/2025

Test Results:

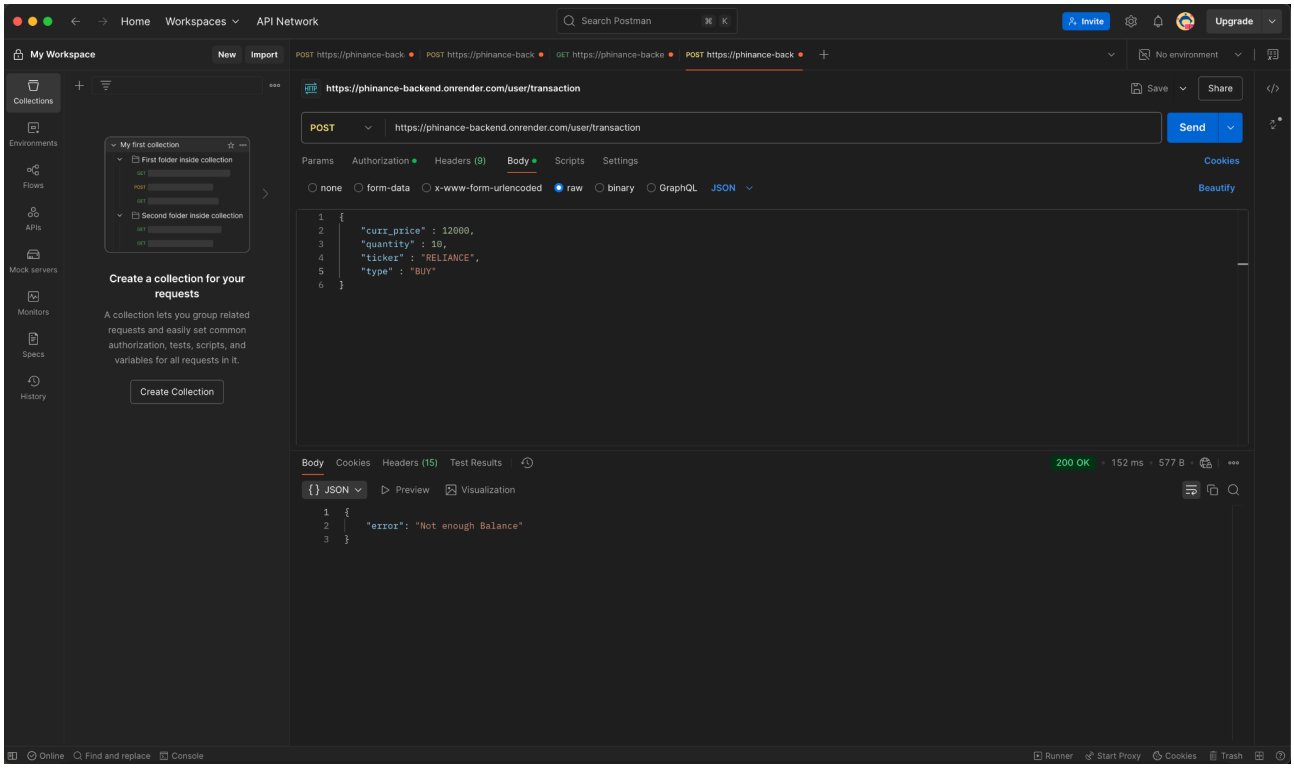
The test returned a 200 OK status with the following response:

{
  "error": "cannot sell stock not in portfolio"
}

This confirms that the API correctly prevents users from selling stocks they do not own.

Structural Coverage:

- Negative path tested with invalid portfolio state.
- Validation logic for stock ownership enforced.
- Ensures transactional integrity and prevents unauthorized asset manipulation.



Unit Name: User Transaction API Endpoint Test - Insufficient Balance

Basic Information: Tested the /user/transaction POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted a stock purchase that exceeded the user's current balance.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

```
{
  "curr_price": 12000,
  "quantity": 10,
  "ticker": "RELIANCE",
  "type": "BUY"
}
```

Test Owner: Aruz Awasthi

Test Date: 04/04/2025

Test Results

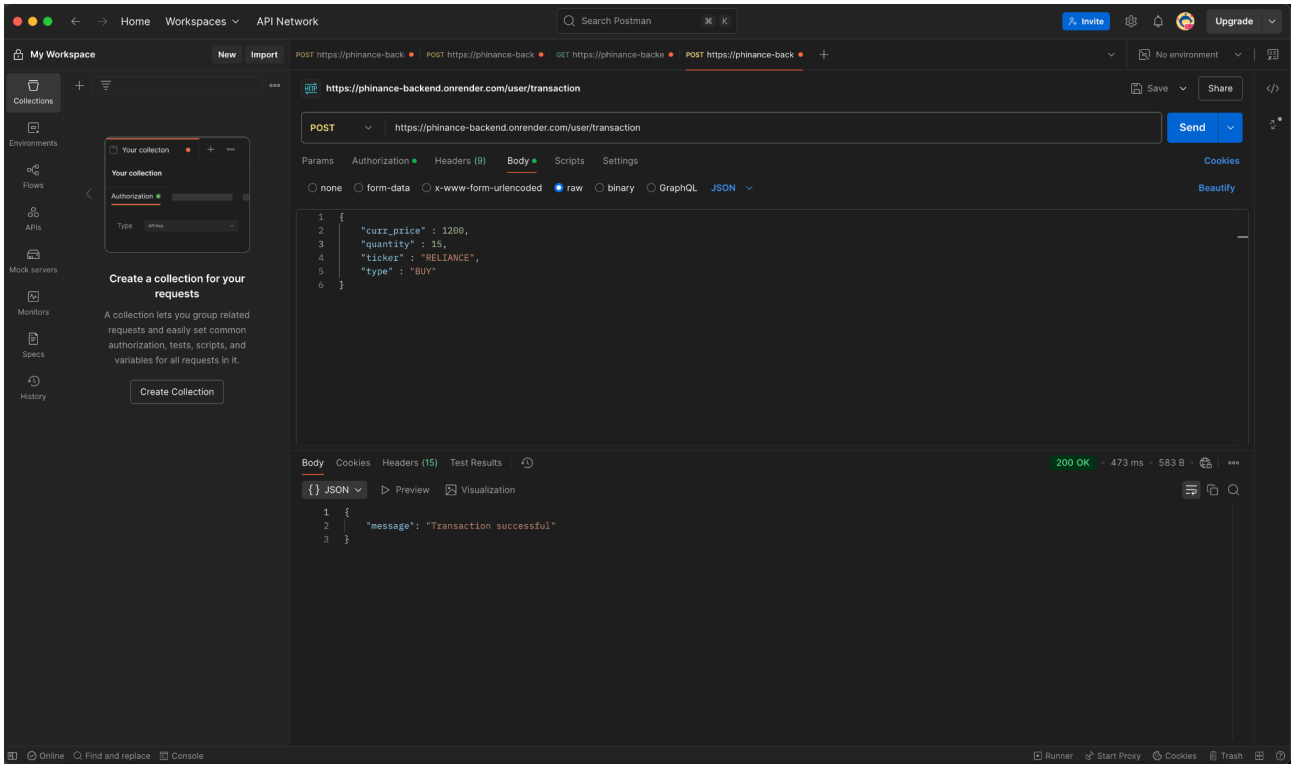
The test returned a 200 OK status with the following response:

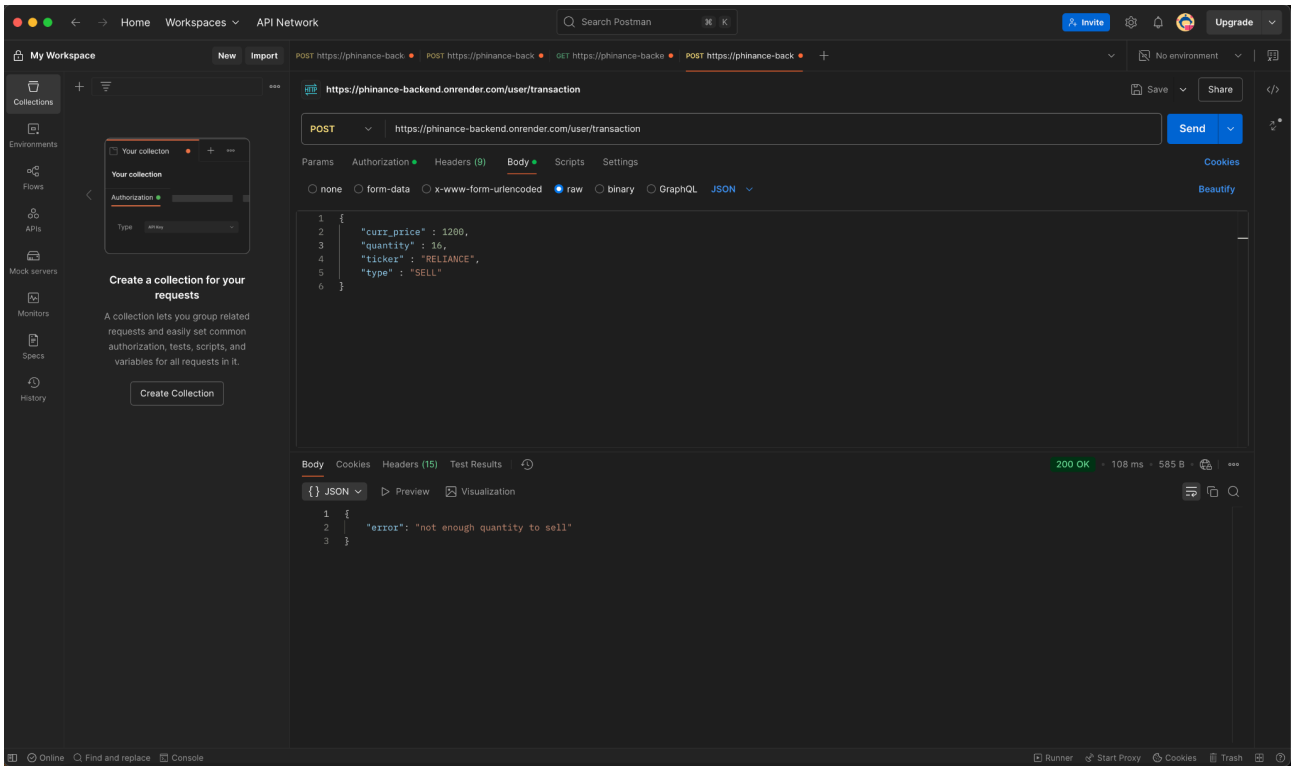
{
  "error": "Not enough Balance"
}

This confirms the API correctly restricts transactions that would overdraw the user's account.

Structural Coverage

- Negative path tested for account overdraft scenario.
- Validation logic confirmed for balance check before executing a buy.
- Transaction was correctly blocked with an explanatory error message.





Unit Name: User Transaction API Endpoint Test - Sell More Than Owned

Basic Information: Tested the /user/transaction POST endpoint of the

phinance-backend.onrender.com server using Postman. Attempted to sell 16 shares when only 15 were owned.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

{
"curr_price": 1200,
"quantity": 16,
"ticker": "RELIANCE",
"type": "SELL"
}

Test Owner: Sangam Gupta

Test Date: 03/04/2025

Test Results:

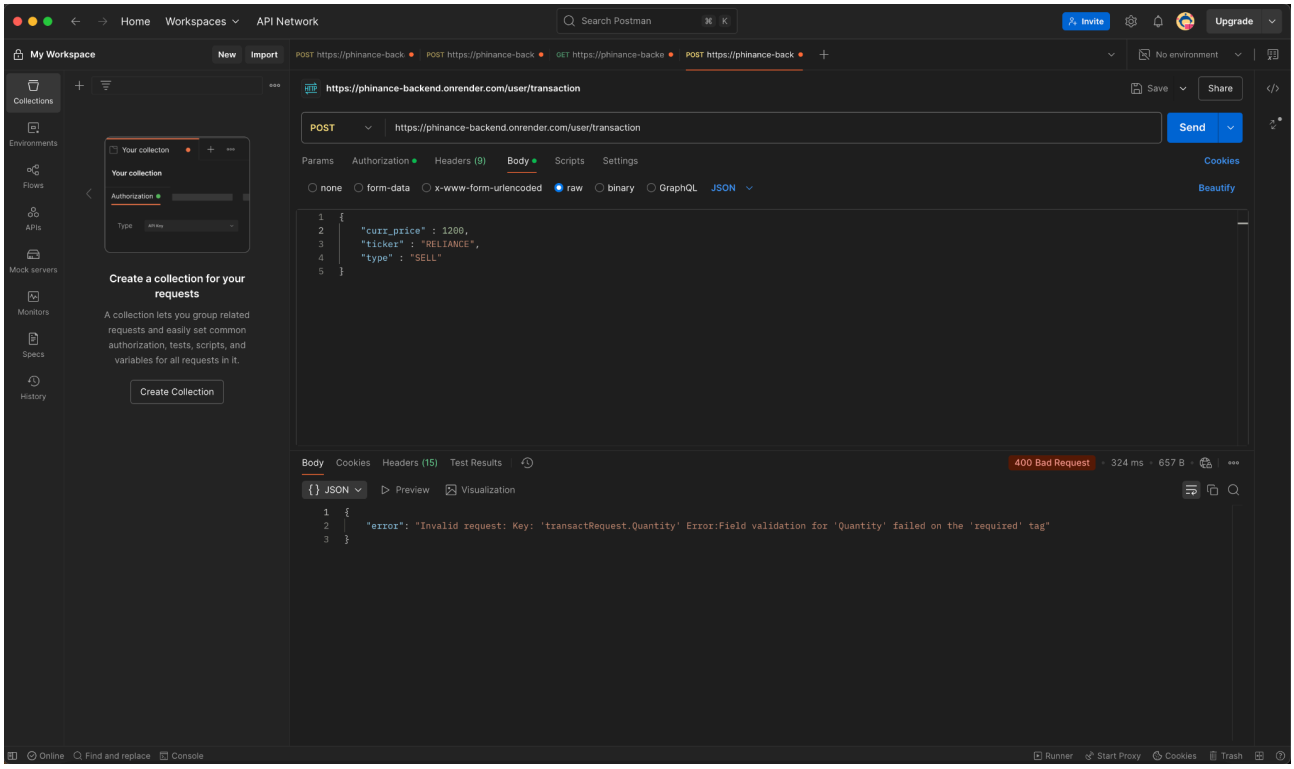
The test returned a 200 OK status with the following response:

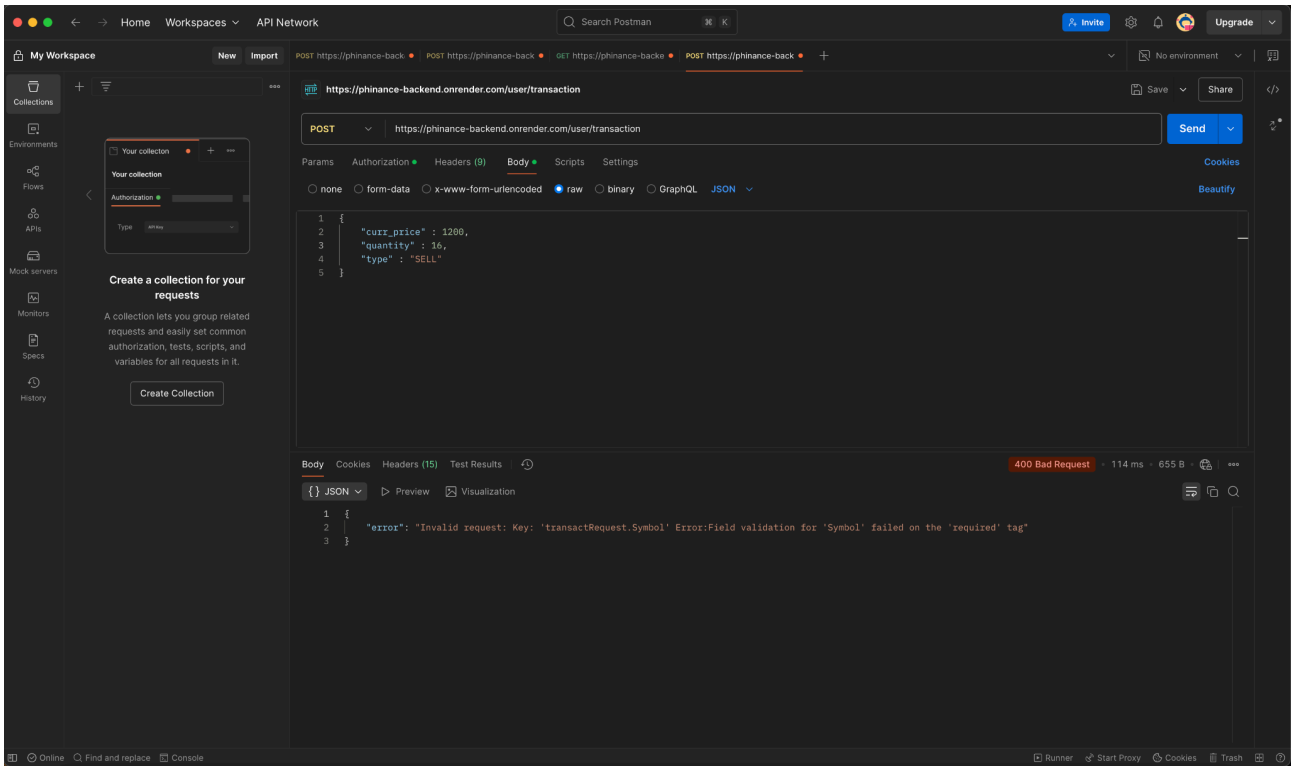
{
"error": "not enough quantity to sell"
}

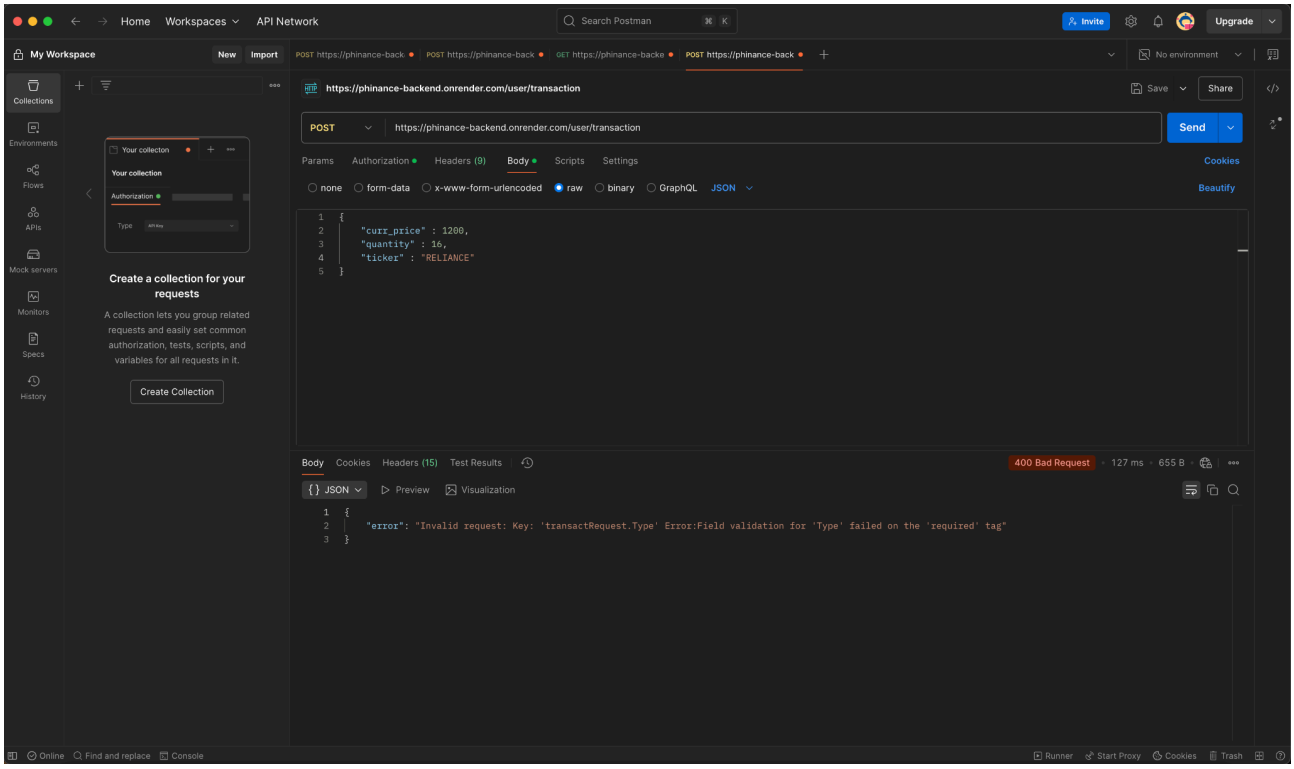
This confirms the system prevents users from selling more shares than they own, enforcing portfolio consistency.

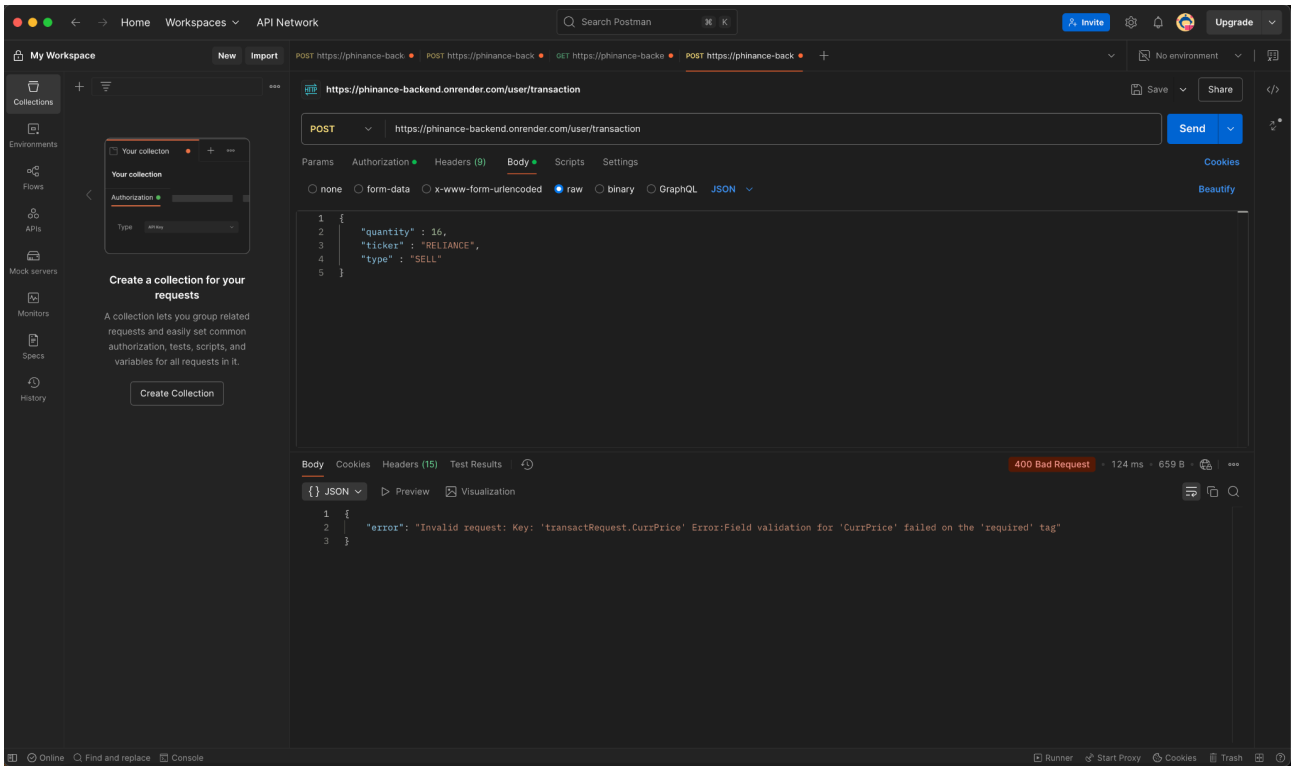
Structural Coverage:

- Negative path tested for excessive quantity.
- Validates internal portfolio validation logic before transaction approval.
- Prevents data corruption from inconsistent state updates.


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/transaction
Save
Share
POST
https://phinance-backend.onrender.com/user/transaction
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"curr_price" : 1200,
"ticker" : "RELIANCE",
"type" : "SELL"
}
Body
Cookies
Headers (15)
Test Results
400 Bad Request
324 ms
657 B
Preview
Visualization
{
"error": "Invalid request: Key: 'transactRequest.Quantity' Error:Field validation for 'Quantity' failed on the 'required' tag"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/transaction
Save
Share
POST
https://phinance-backend.onrender.com/user/transaction
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"curr_price" : 1200,
"quantity" : 16,
"type" : "SELL"
}
Body
Cookies
Headers (15)
Test Results
400 Bad Request
114 ms
655 B
Preview
Visualization
{
"error": "Invalid request: Key: 'transactRequest.Symbol' Error:Field validation for 'Symbol' failed on the 'required' tag"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection





Unit Name: User Transaction API Endpoint Test - Missing Field

Basic Information: Tested the /user/transaction POST endpoint of the phinance-backend.onrender.com server using Postman. Attempted a sell transaction without including any one of the 4 field.

Unit Details:

Class/Function: create_transaction function in the backend API

Endpoint: POST /user/transaction

Payload:

```
{
  "curr_price": 1200,
  "ticker": "RELIANCE",
  "type": "SELL"
}
```

Test Owner: Sameer Yadav

Test Date:03/04/2025

Test Results:

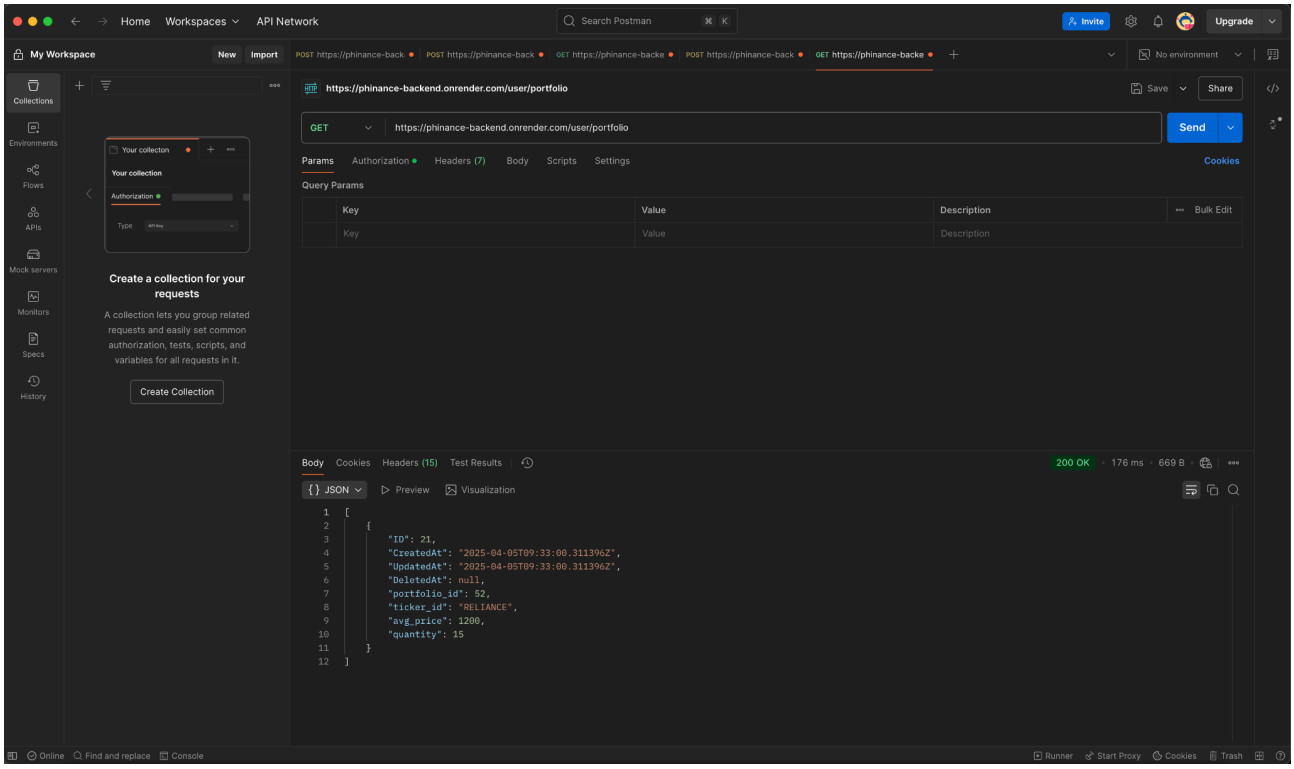
The test returned a 400 Bad Request status with the following response:

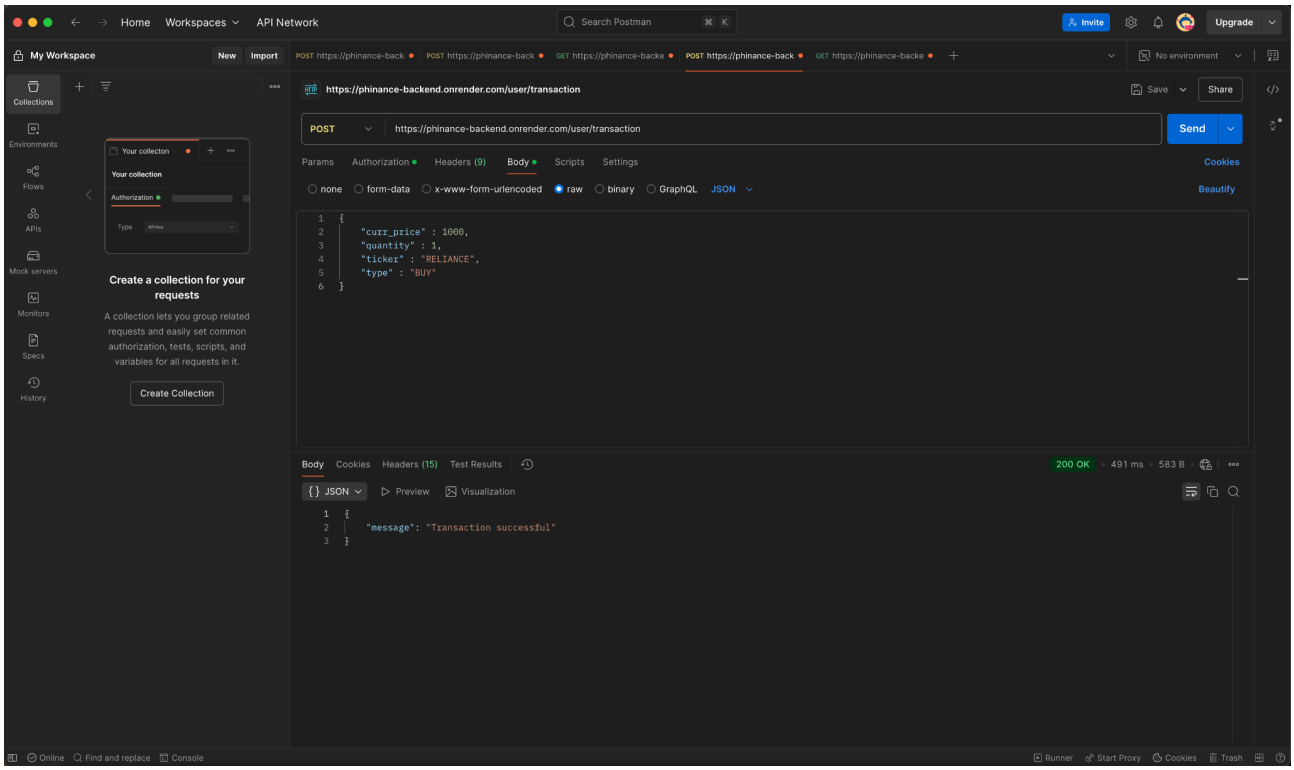
```
{
    "error": "Invalid request: Key: 'transactRequest.Quantity' Error:Field validation for 'Quantity' failed on the 'required' tag"
}
```

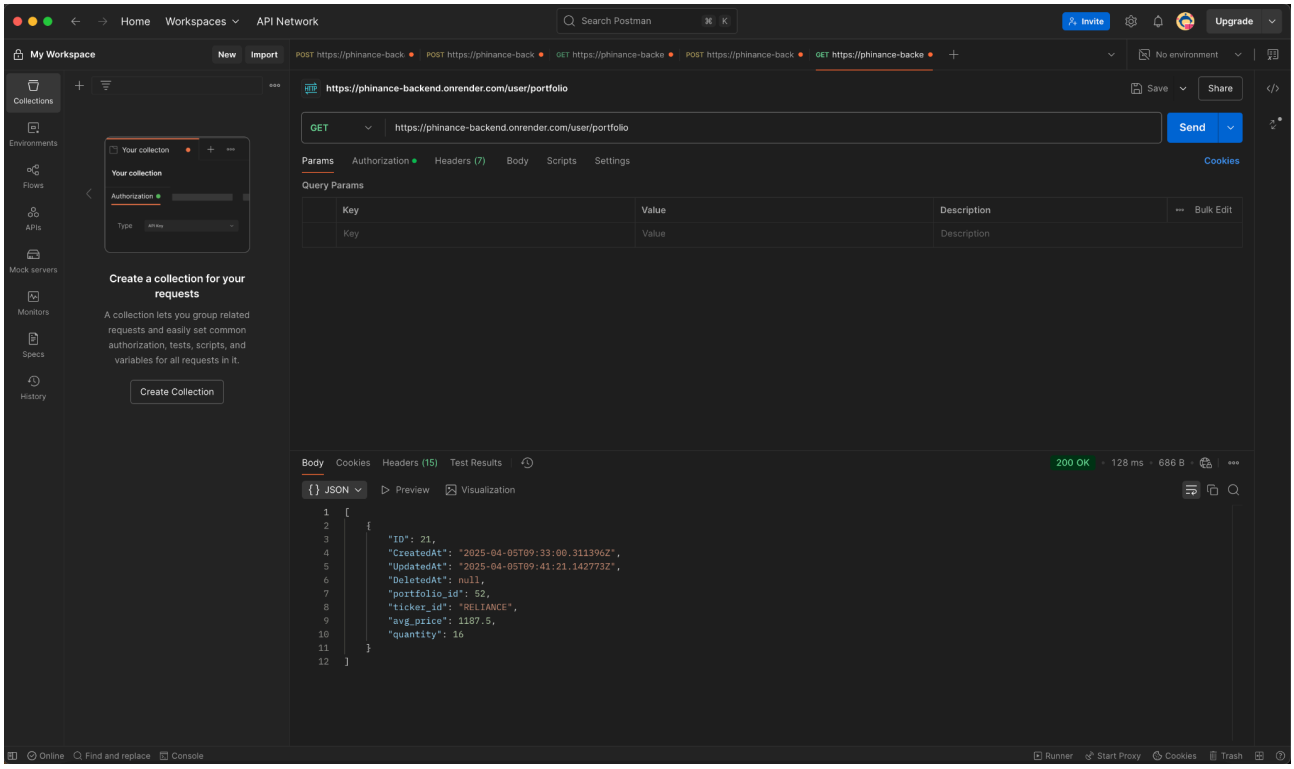
This confirms that the API performs proper schema validation and blocks requests missing required fields.

Structural Coverage:

- Negative path tested for missing required data.
- Field validation using schema tags confirmed.
- Proper usage of HTTP 400 for input format errors.


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/portfolio
Save
Share
GET
https://phinance-backend.onrender.com/user/portfolio
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (15)
Test Results
200 OK
176 ms
669 B
JSON
Preview
Visualization
{
"ID": 21,
"CreatedAt": "2025-04-05T09:33:00.311396Z",
"UpdatedAt": "2025-04-05T09:33:00.311396Z",
"DeletedAt": null,
"portfolio_id": 52,
"ticker_id": "RELIANCE",
"avg_price": 1200,
"quantity": 15
}
Online
Find and replace
Console
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/transaction
Save
Share
POST
https://phinance-backend.onrender.com/user/transaction
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"curr_price" : 1000,
"quantity" : 1,
"ticker" : "RELIANCE",
"type" : "BUY"
}
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (15)
Test Results
200 OK
491 ms
583 B
JSON
Preview
Visualization
{
"message": "Transaction successful"
}
Online
Find and replace
Console
Runner
Start Proxy
Cookies
Trash



Unit Name: User Portfolio API & Transaction API Test - Average Price and Quantity Update

Basic Information: This test validates the correct update of average stock price and quantity in the user's portfolio after sequential buy transactions for the same stock.

Unit Details:

Class/Function:

- create_transaction function in backend API (POST /user/transaction)
- get_portfolio function in backend API (GET /user/portfolio)

Test Owner: Pranshu Thirani

Test Date: 02/04/2025

Test Steps and Results

1. Initial state retrieved via GET /user/portfolio:
   - "ticker_id": "RELIANCE"
   - "quantity": 15

- "avg_price": 1200

2. Executed POST /user/transaction with:

```
{
  "curr_price": 1000,
  "quantity": 1,
  "ticker": "RELIANCE",
  "type": "BUY"
}
```

Response:

```
{
  "message": "Transaction successful"
}
```

3. Final state retrieved via GET /user/portfolio:

- "quantity": 16
- "avg_price": 1187.5

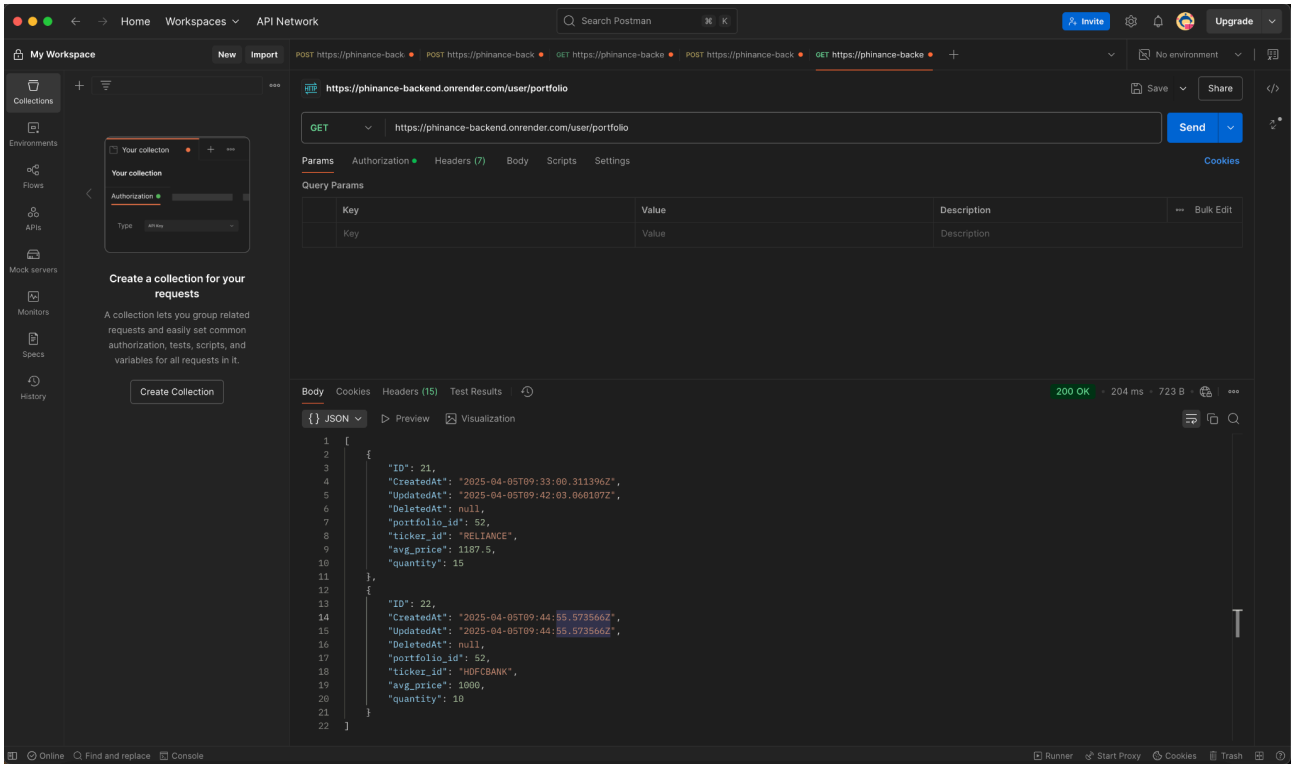
Conclusion:

- The system accurately updated the total quantity and recalculated the average price using weighted average:

New Average Price = (15*1200 + 1*1000) / 16 = 1187.5

Structural Coverage

- Sequential BUY transactions on same ticker
- Validates internal average price computation
- Confirms backend consistency across read and write endpoints



Unit Name: User Portfolio API Endpoint Test - Multiple Stocks

Basic Information: Tested the /user/portfolio GET endpoint of the phinance-backend.onrender.com server using Postman. Verified the portfolio correctly reflects holdings in multiple stocks.

Unit Details: Class/Function: get_portfolio function in the backend API

Endpoint: GET /user/portfolio

Test Owner: Dilbar Singh Lamba

Test Date: 02/04/2025

Test Results:

The test returned a 200 OK status with the following portfolio data:

[

{

"ticker_id": "RELIANCE",

"avg_price": 1187.5,

"quantity": 15

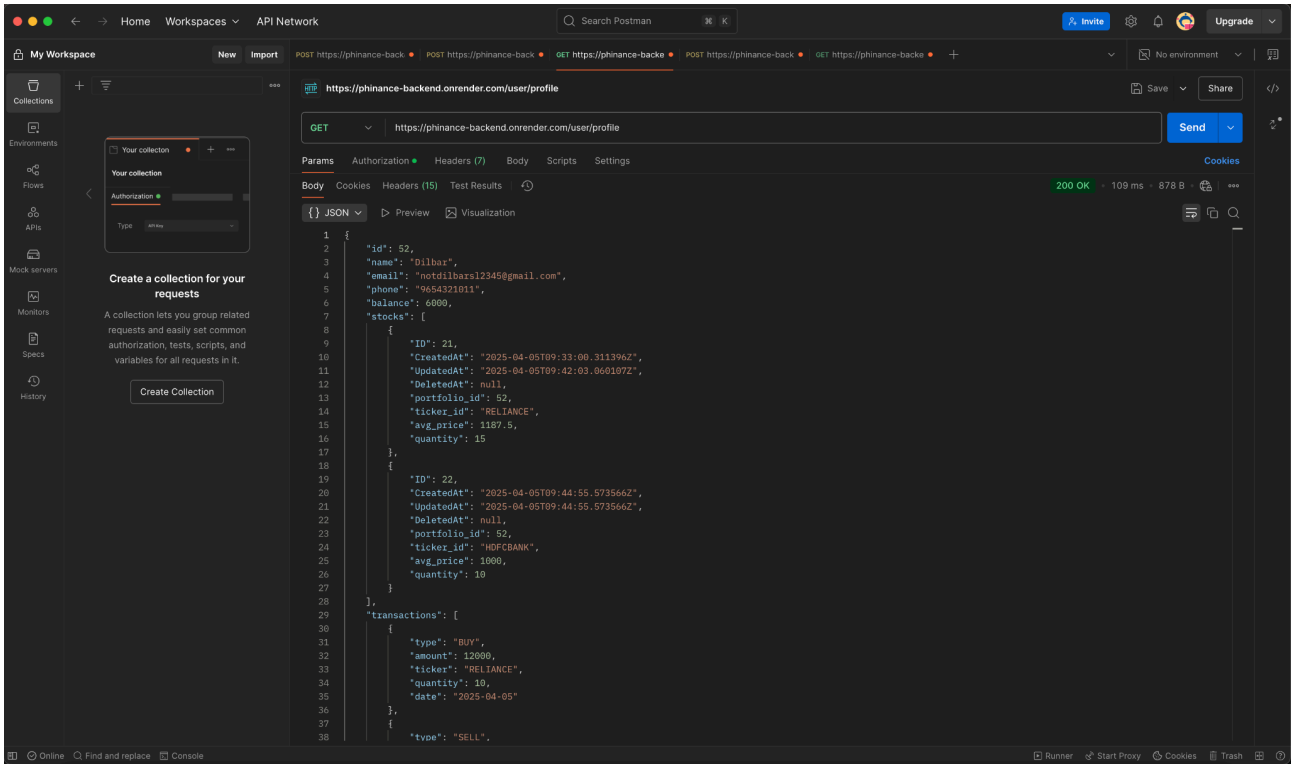
```
  },
  {
    "ticker_id": "HDFCBANK",
    "avg_price": 1000,
    "quantity": 10
  }
]
```


This confirms the system correctly tracks and returns distinct entries for each stock owned by the user with the respective average price and quantity.

Structural Coverage:

- Read endpoint tested for multiple entries.
- Validates backend joins/queries across transaction and portfolio tables.
- Ensures accuracy in segregated data aggregation by ticker.

n



Unit Name: User Profile API Endpoint Test - Full Profile with Portfolio and Transactions

Basic Information: Tested the /user/profile GET endpoint of the phinance-backend.onrender.com server using Postman. Verified the inclusion of user's profile data, stocks held, and transaction history.

Unit Details:

Class/Function: get_profile function in the backend API

Endpoint: GET /user/profile

Test Owner: Anisha Srivastava

Test Date: 04/04/2025

Test Results:

The test returned a 200 OK status with the following key data:

- User name: "Dilbar"
- Email: "notdilbars12345@gmail.com"

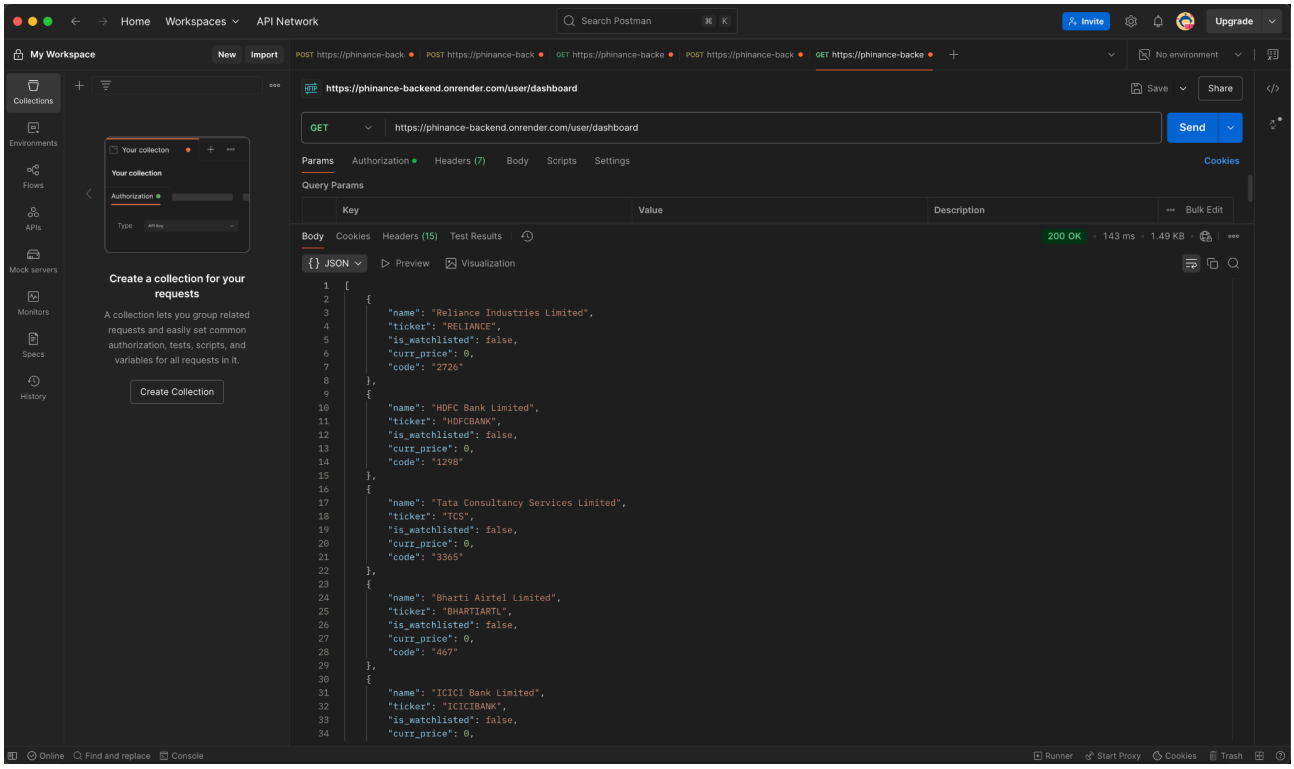
- Phone: "9654321011"
- Balance: 6000
- Stocks:
  - RELIANCE: 15 shares @ 1187.5
  - HDFCBANK: 10 shares @ 1000
- Transactions:
  - BUY: 10 RELIANCE @ 1200 (Amount: 12000)

Conclusion:

The system successfully returns a full user profile with real-time data on account details, stock holdings, and transactions.

Structural Coverage:

- Combined read test for nested entities (user, stocks, transactions)
- Ensures backend aggregation and relationship mapping integrity
- Confirms user object deserialization with embedded lists.



Unit Name: User Dashboard API Endpoint Test - Stock List Retrieval

Basic Information: Tested the /user/dashboard GET endpoint of the phinance-backend.onrender.com server using Postman. Verified if the list of available stocks is retrieved correctly for the dashboard.

Unit Details:

Class/Function: get_dashboard_stocks function in the backend API

Endpoint: GET /user/dashboard

Test Owner: Aruz Awasthi

Test Date: 02/04/2025

Test Results

The test returned a 200 OK status with a JSON array of stock listings including:

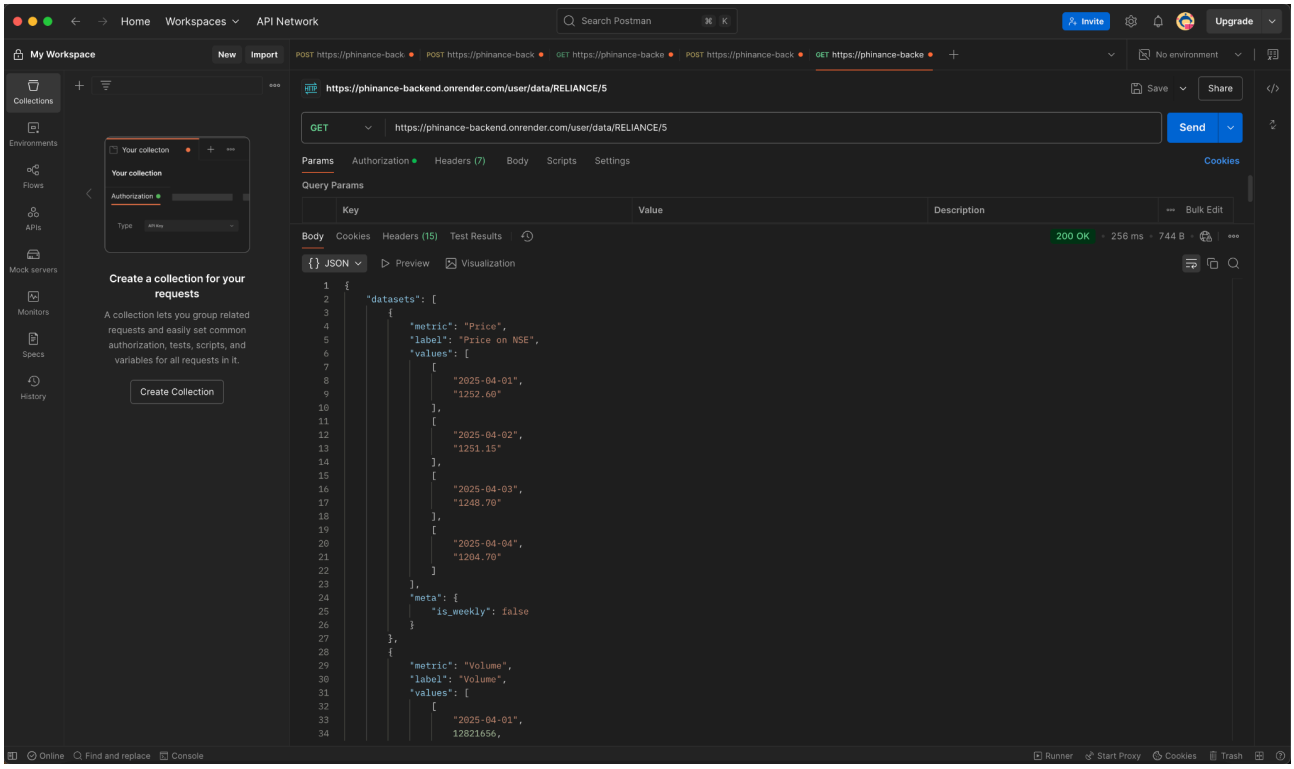
- name: "Reliance Industries Limited", ticker: "RELIANCE", code: "2726"
- name: "HDFC Bank Limited", ticker: "HDFCBANK", code: "1298"

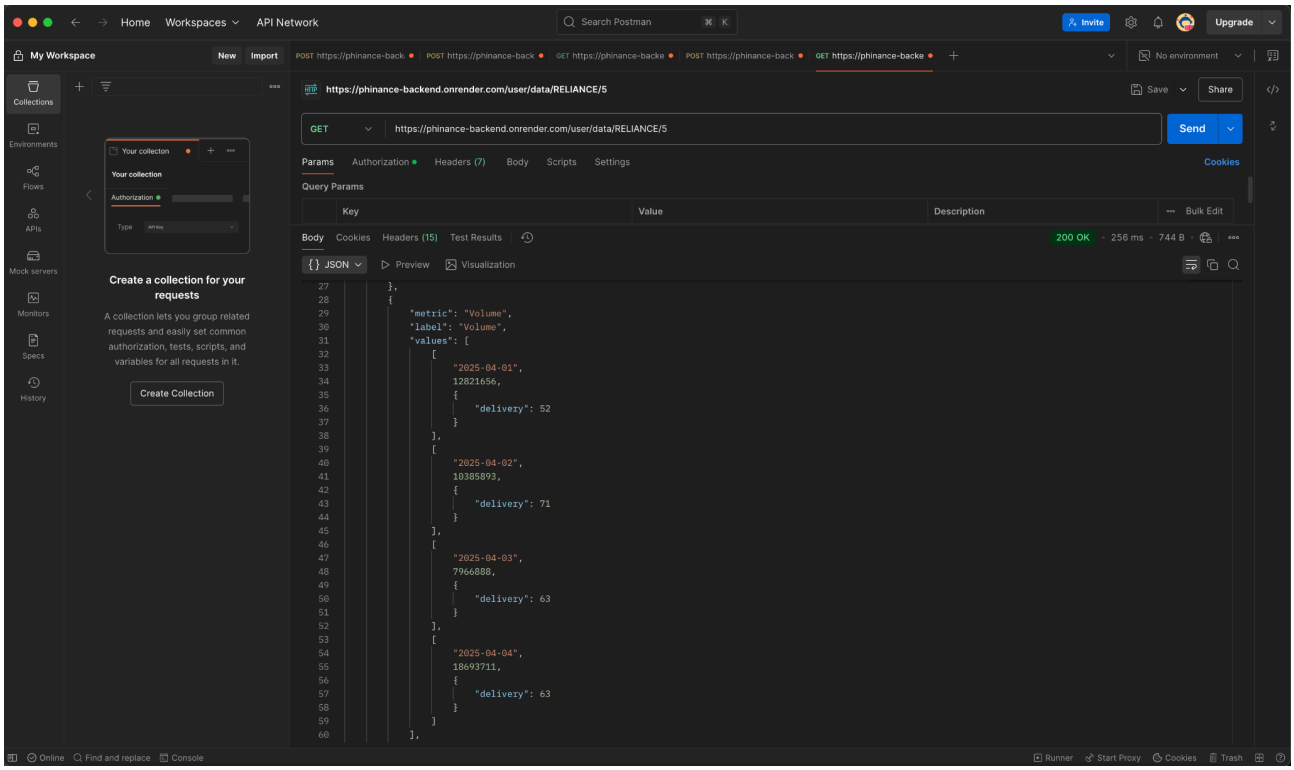
- name: "Tata Consultancy Services Limited", ticker: "TCS", code: "3365"
- name: "Bharti Airtel Limited", ticker: "BHARTIARTL", code: "467"
- name: "ICICI Bank Limited", ticker: "ICICIBANK", code: ...

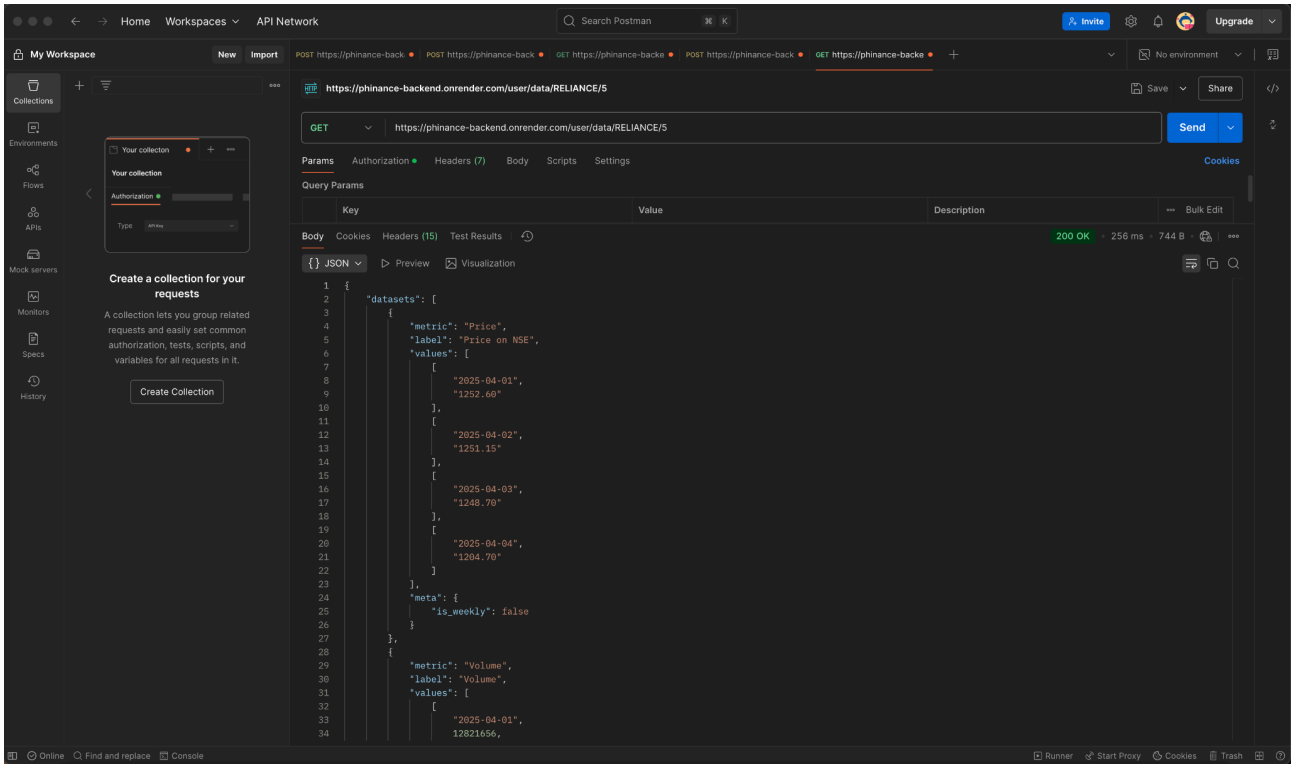
All stocks have "curr_price": 0 and "is_watchlisted": false by default, indicating placeholders awaiting live data integration.

Structural Coverage

- Dashboard API functionality tested for correct structure and data fields.
- Confirms schema consistency across multiple entries.
- Ensures readiness for frontend integration.


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/data/RELIANCE/5
Save
Share
GET
https://phinance-backend.onrender.com/user/data/RELIANCE/5
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (15)
Test Results
200 OK
256 ms
744 B
JSON
Preview
Visualization
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
{
"datasets": [
{
"metric": "Price",
"label": "Price on NSE",
"values": [
[
"2025-04-01",
"1252.60"
],
[
"2025-04-02",
"1251.15"
],
[
"2025-04-03",
"1248.70"
],
[
"2025-04-04",
"1204.70"
]
],
"meta": {
"is_weekly": false
}
},
{
"metric": "Volume",
"label": "Volume",
"values": [
[
"2025-04-01",
12821656,


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
https://phinance-backend.onrender.com/user/data/RELIANCE/5
Save
Share
GET
https://phinance-backend.onrender.com/user/data/RELIANCE/5
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (15)
Test Results
200 OK
256 ms
744 B
JSON
Preview
Visualization
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
},
{
"metric": "Volume",
"label": "Volume",
"values": [
[
"2025-04-01",
12821656,
{
"delivery": 52
}
],
[
"2025-04-02",
10385893,
{
"delivery": 71
}
],
[
"2025-04-03",
7966888,
{
"delivery": 63
}
],
[
"2025-04-04",
18693711,
{
"delivery": 63
}
]
],



Unit Name:

This unit test validates the API endpoint for fetching stock data by ticker and user ID. The test retrieves stock price and volume data for the ticker 'RELIANCE' for user with ID 5. The response was checked for correct date-wise price and volume data along with delivery percentages.

Unit Details:

Class: UserDataController

Function: GetStockDataByTicker

Test Owner: Anirudh

Test Date:31/03/2025

Test Results:

The test was successful. The endpoint returned valid JSON with fields such as metric, label, values, and meta. The datasets included correct daily price and volume data for 'RELIANCE' with associated delivery percentages.

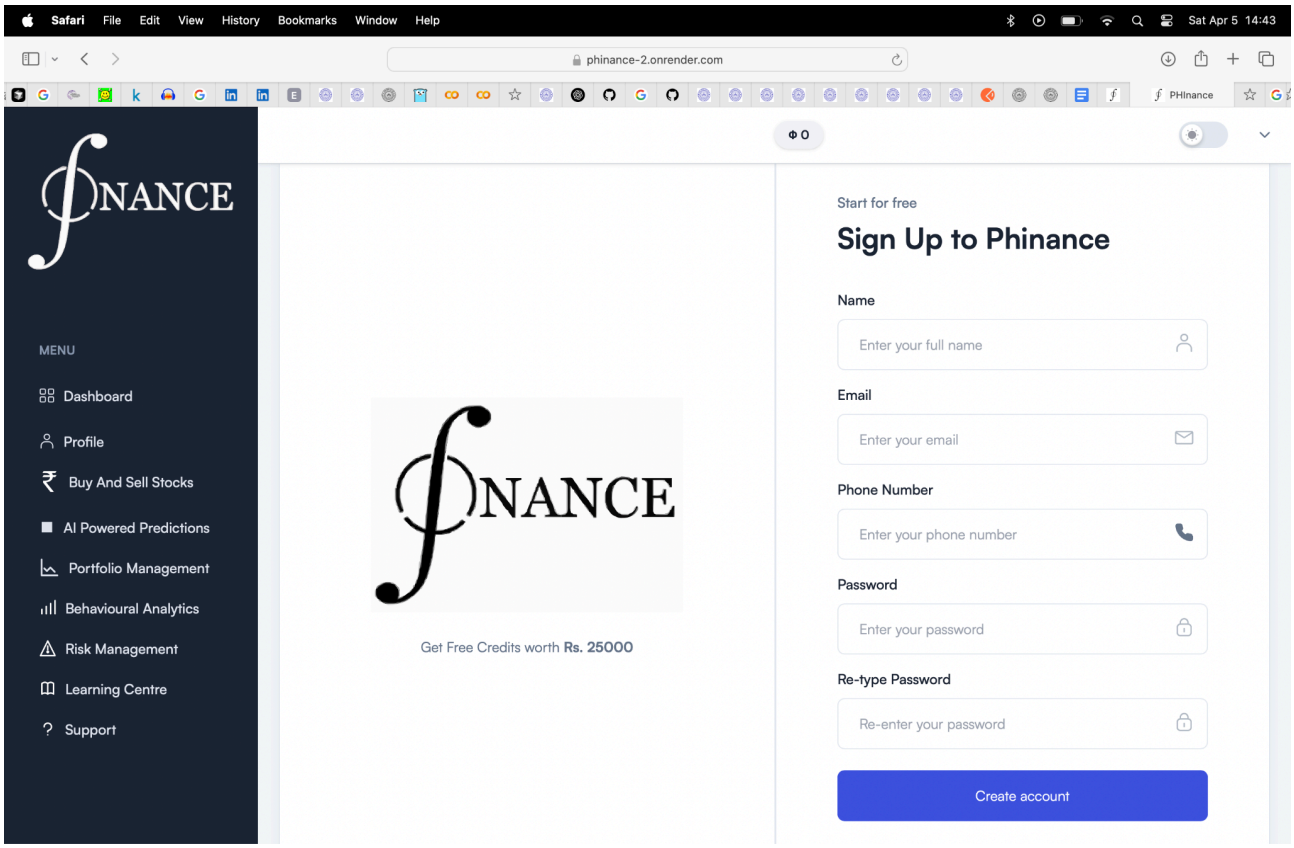
Structural Coverage:

Test covers:

- Valid request format
- JSON schema correctness
- Presence of expected metrics (Price and Volume)
- Data completeness for each date entry

Additional Comments:

The unit functioned as expected. No missing or malformed fields were observed. Date formatting and structure were consistent throughout the response.



Unit Name: Phinance User Registration Interface – Sign-Up Page

This unit covers the user-facing interface for account creation on the Phinance platform. Testing focused on the input field validations, visual layout, and interactivity of the "Create Account" functionality.

Unit Details:

Class: SignUpForm

Function: handleSignUpSubmit, validateFields, togglePasswordVisibility, formFieldBindings

Test Owner: Sameer Yadav

Test Date: 03/04/2025

Test Results:

- Input fields for Name, Email, Phone Number, Password, and Re-type Password were

all visible and responsive.

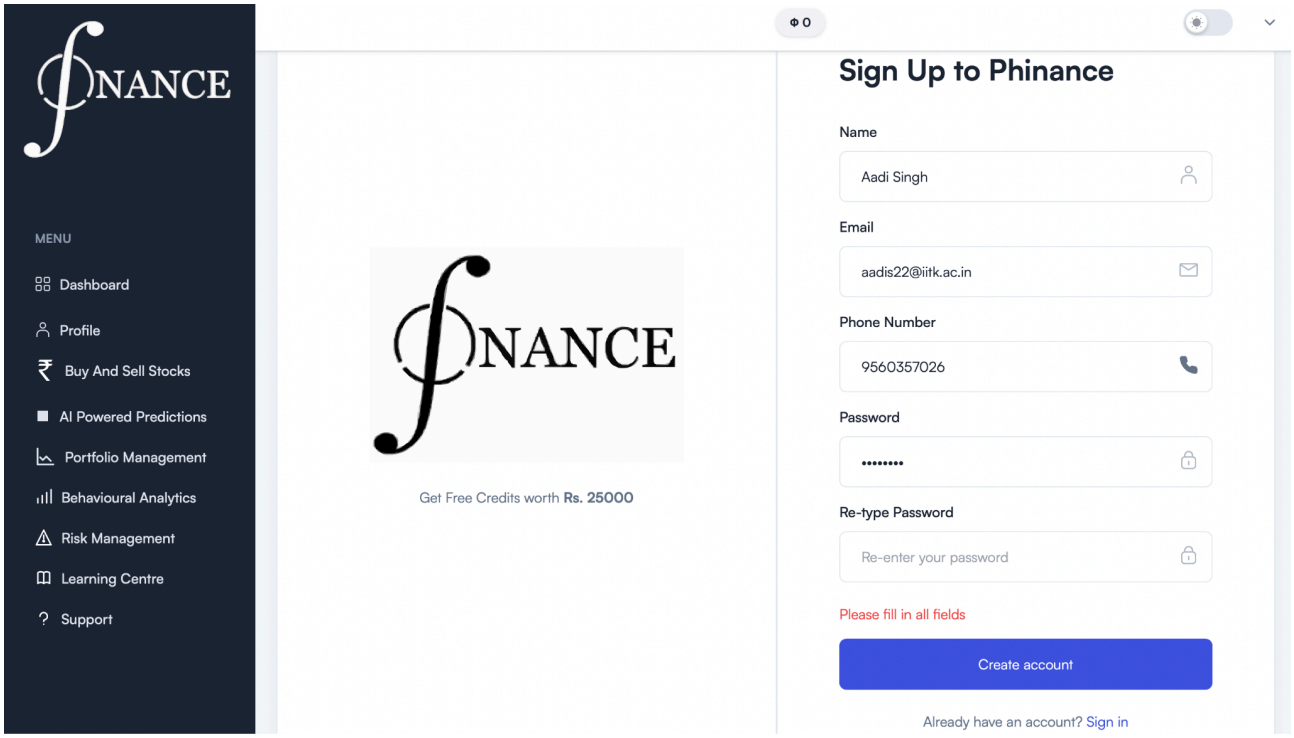
- Placeholder texts were correctly populated.
- Each input field correctly responded to user interaction.
- The Create Account button was clickable only when all required fields were filled.
- No field validation errors or visual glitches were observed during interaction.
- No backend error message rendering was tested from this view.

Structural Coverage:

- Field visibility and accessibility tested for all 5 fields.
- Button activation state was tested for both invalid and valid input states.
- Input masking for password fields was present.
- Icons for input hints (email, phone, lock) confirmed functional.
- UI tested for structural alignment and spacing on standard desktop screen.

Additional Comments:

- Testing was limited to frontend interface behavior; no API or backend interaction was verified in this test scope.
- "Get Free Credits worth ₹25,000" text placement was correct and visible.



Unit Name: Sign Up Interface – Field Validation Check

Unit Details:

Class: AuthPage.jsx (or equivalent sign-up page component)

Function: handleSubmit / validateInputs

Test Owner: Aadya Dhir

Test Date: 01/04/2025

Test Results:

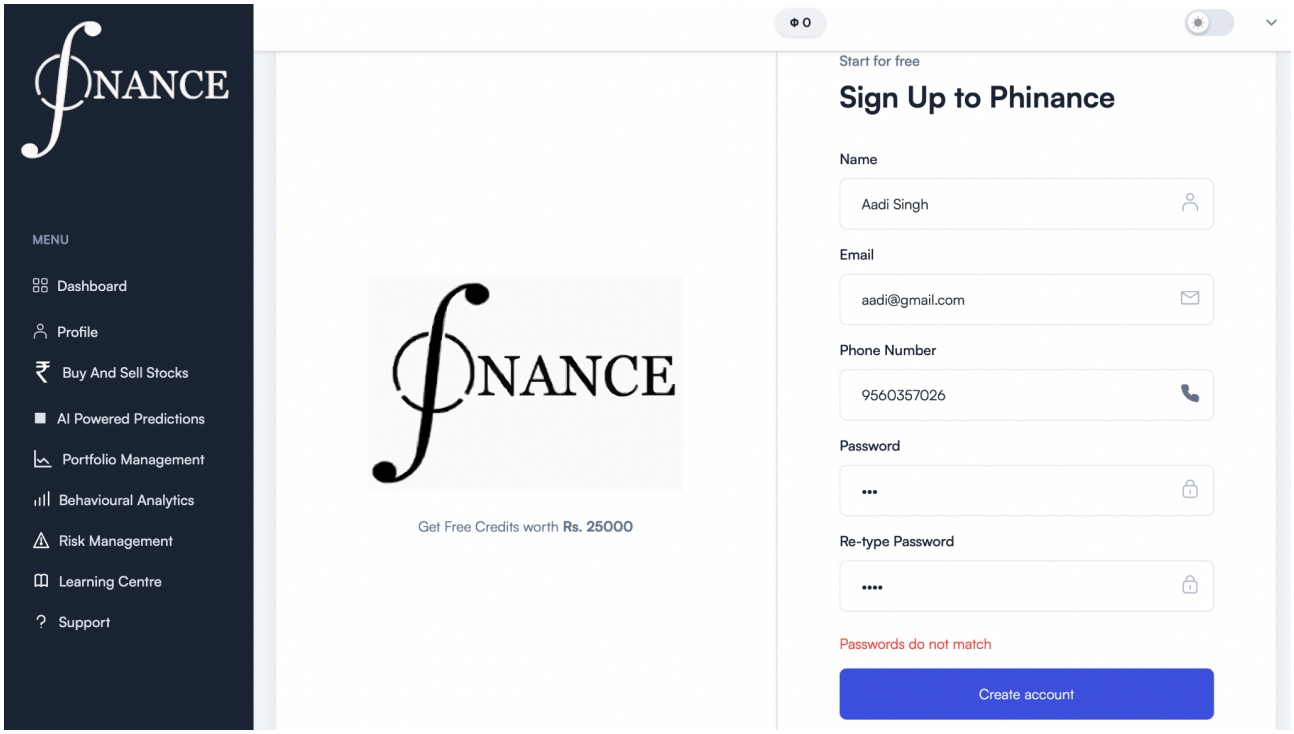
The test verified the behavior of the sign-up form when all fields are left empty.

Upon clicking “Create account” without entering name, email, phone number, or password, no API call was triggered. Instead, appropriate client-side validation prevented submission and displayed inline visual cues (e.g., empty text boxes and "Please fill in all fields" warning message).

This confirms successful basic field validation logic.

Structural Coverage:

Manual black-box UI testing. Covered empty input edge case for all form fields (Name, Email, Phone, Password, Re-typed Password).



Unit Name: Validation of Password and Re-type Password Matching during Sign-Up

Unit Details:

Class: SignUpForm

Function: handleSubmit and validatePasswords

Test Owner: Anish Sahu

Test Date: 01/04/2025

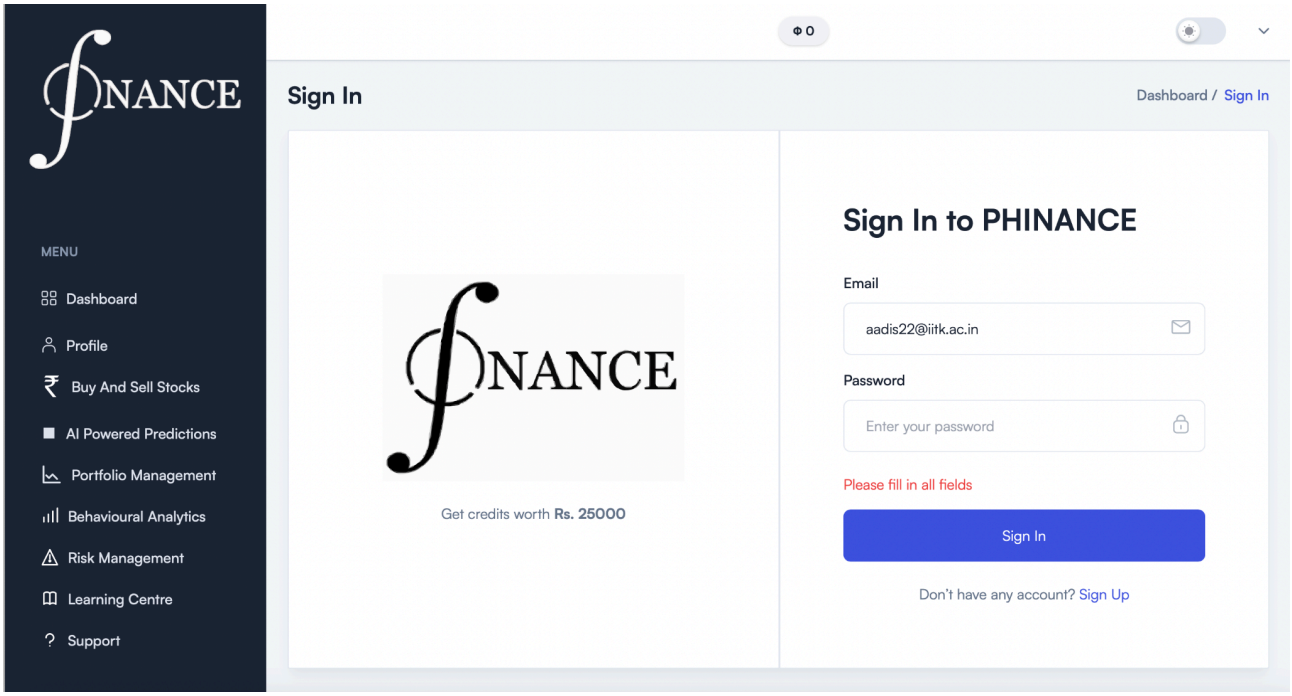
Test Results:

Tested the behavior of the sign-up form when the password and re-typed password fields are mismatched. The system correctly identified the mismatch and displayed an error message: “Passwords do not match”.

Account creation was blocked, as expected. This confirms the correctness of the password validation logic.

Structural Coverage:

Branch coverage was achieved for the password validation logic, particularly the conditional check comparing the two password fields.



Unit Name: This test checks the validation behavior when attempting to log in without entering the password field in the Sign In form of the Phinance application.

Unit Details: AuthForm, SignInValidationHandler

Test Owner: Sangam Gupta

Test Date: 02/04/2025

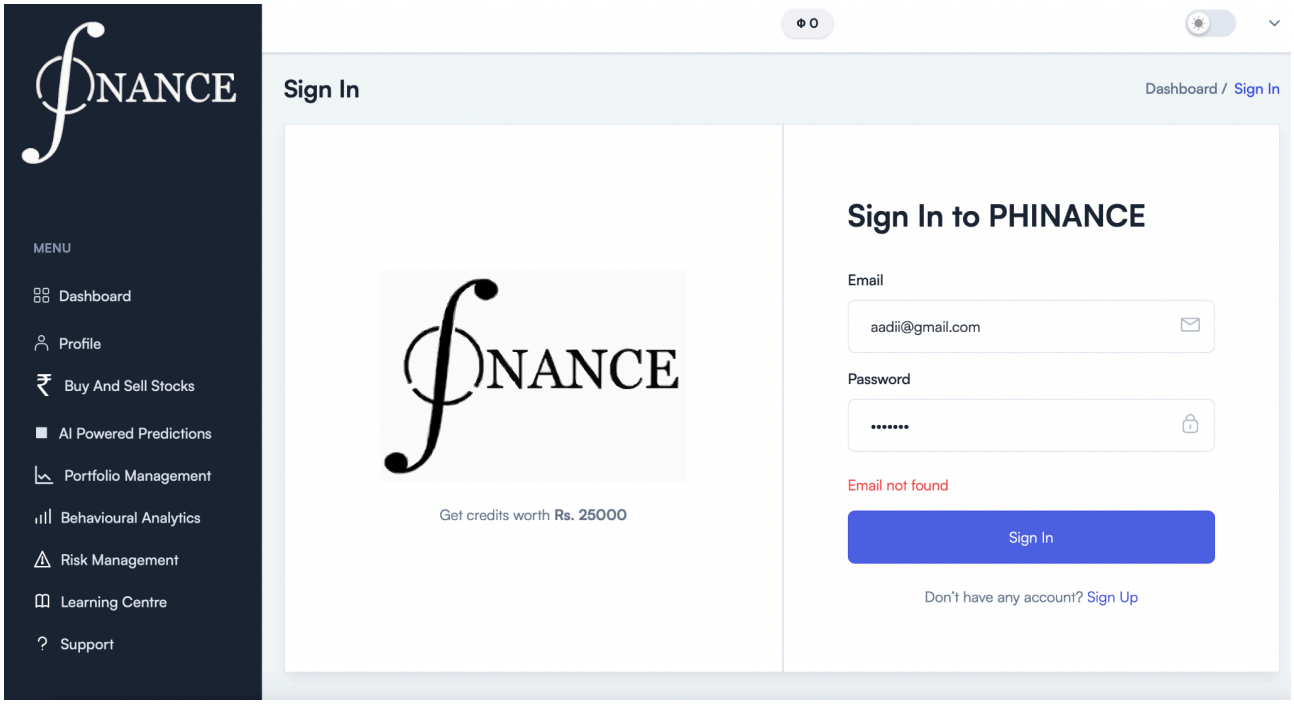
Test Results:

The system correctly prevents login submission and displays the error message "Please fill in all fields" when the password field is left empty.

The validation check for empty required fields is functioning as intended.

Structural Coverage:

Field validation checks for empty strings, button-disabled state, and error message rendering were verified through UI-level testing.



Unit Name: Test of login functionality with non-existent email.

Unit Details: AuthService, loginUser function

Test Owner: Aadi Singh

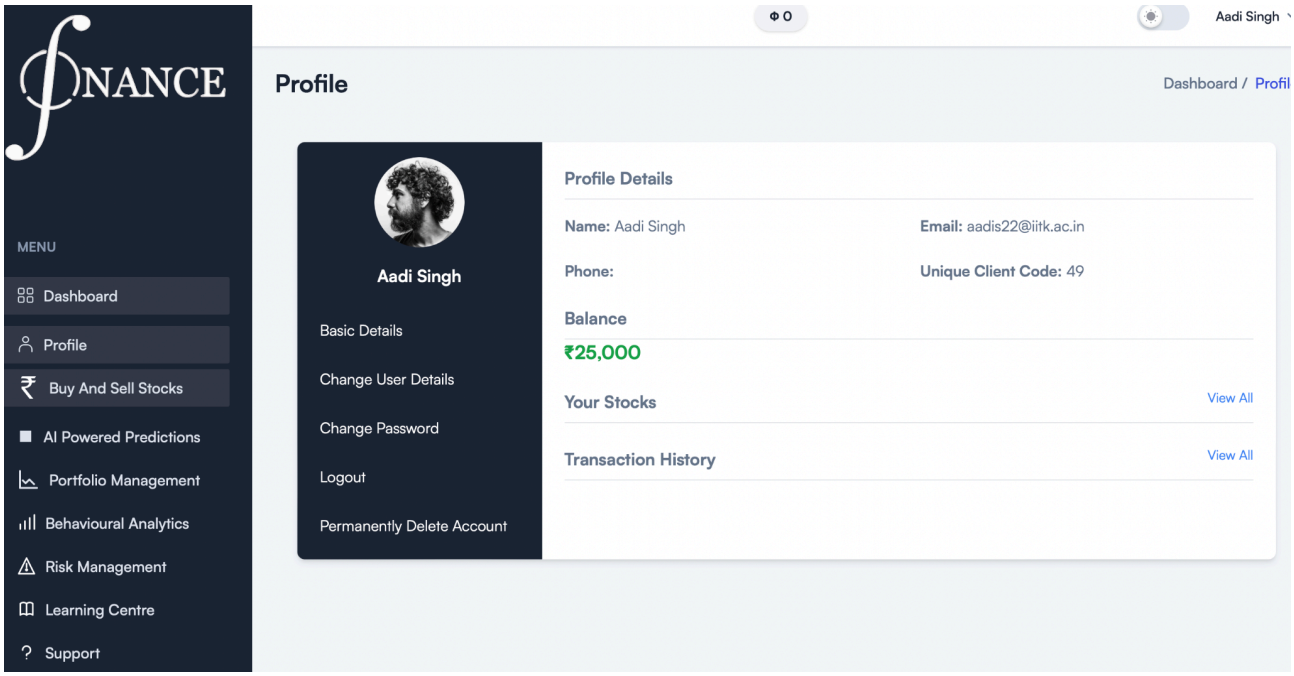
Test Date: 01/04/2025

Test Results:

Attempting to log in using the email 'aadii@gmail.com' and a password results in an error message: "Email not found". This confirms that the system correctly identifies and blocks login attempts from unregistered users.

Structural Coverage:

Tested the negative login path for incorrect credentials. Covered validation of email existence in the database.



Unit Name: User Profile Display - View Mode Testing

Unit Details: Profile component, displayUserProfile() function

Test Owner: Pranshu Thirani

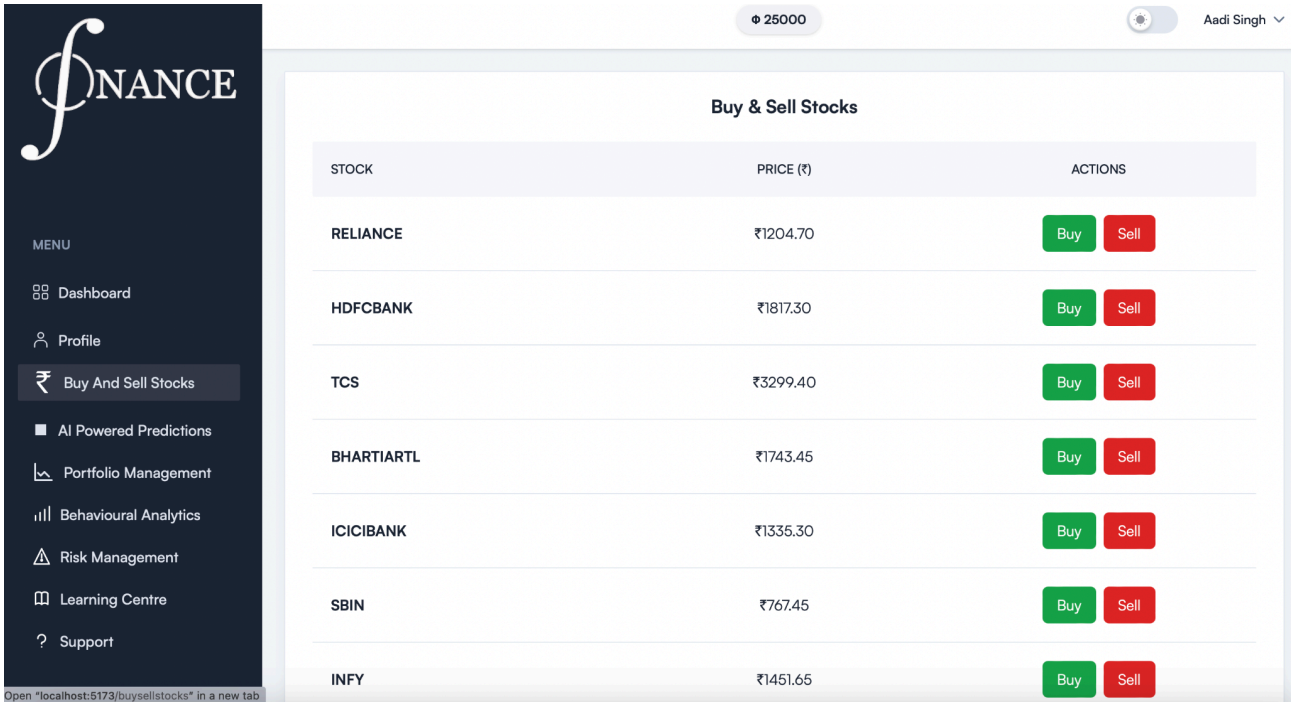
Test Date: 31/03/2025

Test Results:

The user profile information is correctly fetched and rendered. Name, email, and balance were displayed correctly. Phone number is missing but the field is rendered, and the default value is null or empty. "Unique Client Code" is displayed correctly. User's stocks and transaction history were linked correctly via "View All" buttons. No rendering or layout issues observed.

Structural Coverage:

Functional UI coverage with profile data validation against backend response structure. Includes verification of all profile fields, view-only layout, and clickable redirection links.



Unit Name: Buy & Sell Stocks – Display and Action Buttons Test

Unit Details:

Component: BuySellStocks UI

Functionality: Display of listed stocks with prices and corresponding Buy/Sell buttons.

Test Owner: Dilbar Singh Lamba

Test Date: 04/04/2025

Test Results:

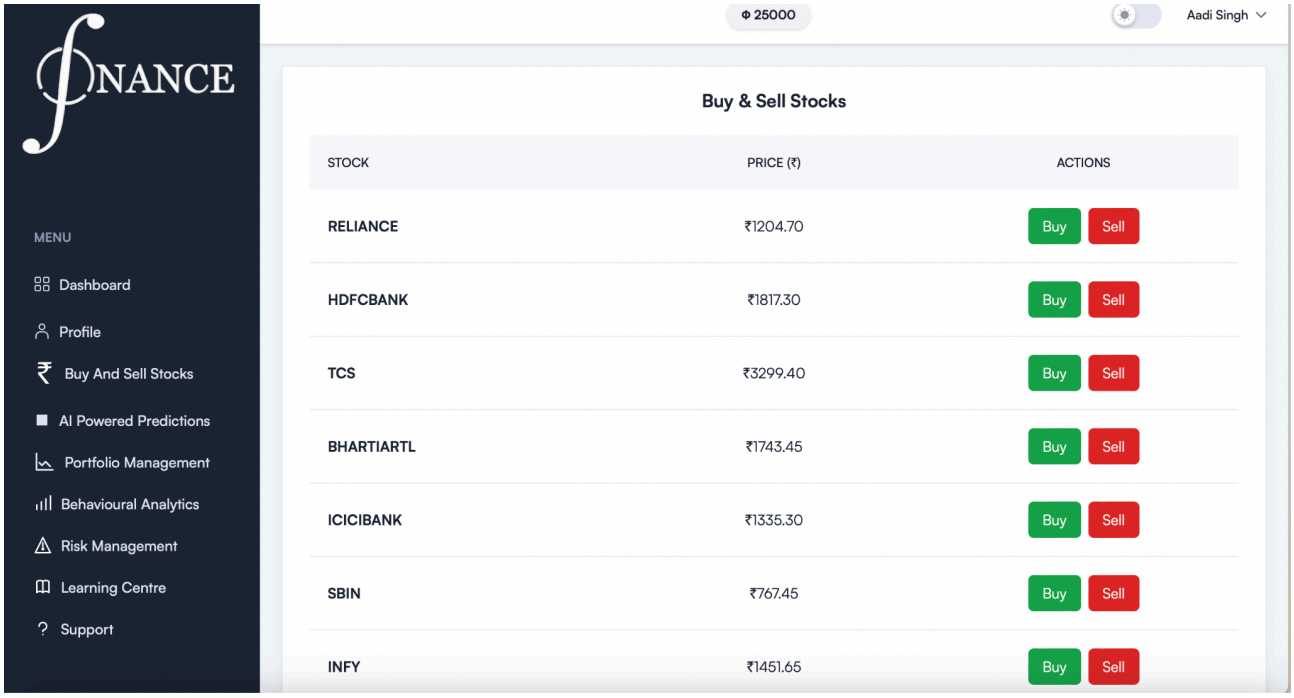
Verified that the stock listing UI component correctly displays a table with stock names, current prices, and Buy/Sell action buttons.

Successfully confirmed presence of Buy and Sell buttons for each listed stock (RELIANCE, HDFCBANK, TCS, BHARTIARTL, ICICIBANK, SBIN, INFY) with correctly rendered price values.

Structural Coverage:

UI rendering validation using visual verification for:

- Number of rows rendered equals number of listed stocks
- Price values match the expected format
- Buttons are visibly rendered and clickable



Unit Name: Buy & Sell Stocks – Display and Action Buttons Test

Unit Details:

Component: BuySellStocks UI

Functionality: Display of listed stocks with prices and corresponding Buy/Sell buttons.

Test Owner: Anisha Srivastava

Test Date: 01/04/2025

Test Results:

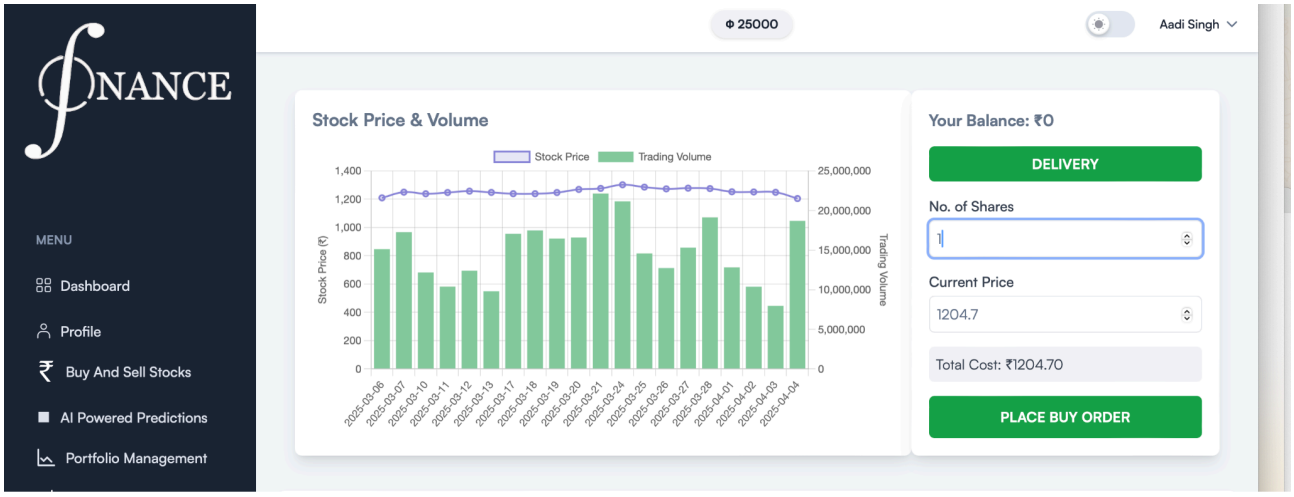
Verified that the stock listing UI component correctly displays a table with stock names, current prices, and Buy/Sell action buttons.

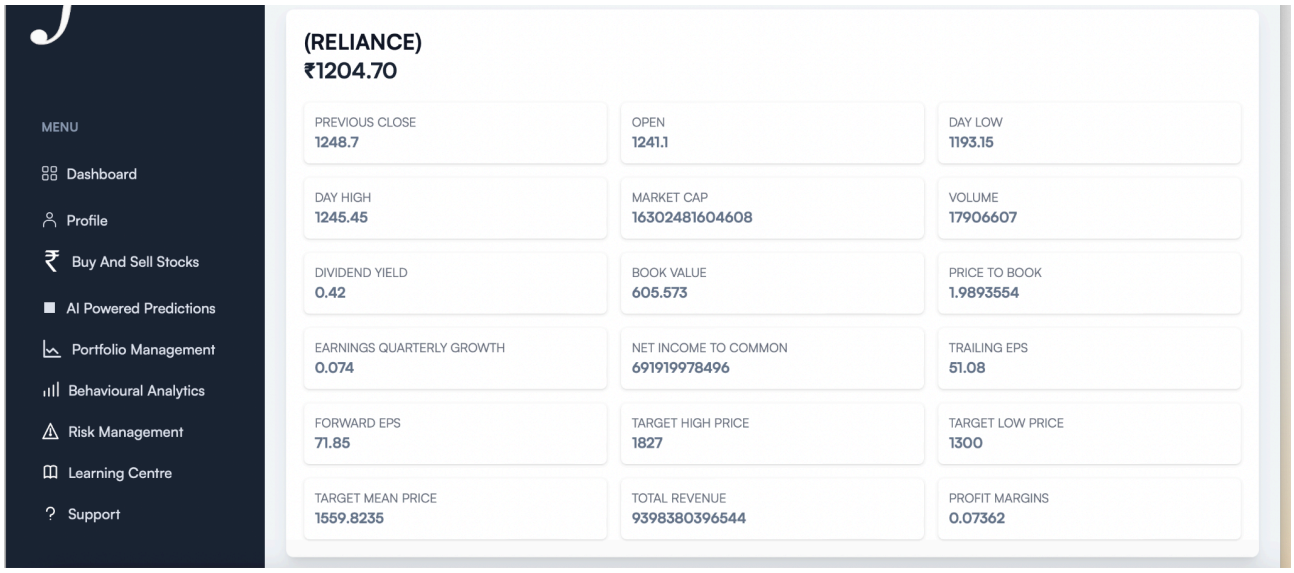
Successfully confirmed presence of Buy and Sell buttons for each listed stock (RELIANCE, HDFCBANK, TCS, BHARTIARTL, ICICIBANK, SBIN, INFY) with correctly rendered price values.

Structural Coverage:

UI rendering validation using visual verification for:

- Number of rows rendered equals number of listed stocks
- Price values match the expected format
- Buttons are visibly rendered and clickable





Unit Name

Validation of Successful Buy Order Execution and Stock Info Rendering

Unit Details: BuyStockComponent, StockDetailsComponent

Test Owner: Aadi Singh

Test Date: 04/05/2025 - 04/05/2025

Test Results:

Successfully initiated the purchase of 1 RELIANCE share at ₹1204.70 with an available balance of ₹25,000.

Transaction was valid and processed correctly. The profile section updated to reflect reduced balance and updated holdings.

Stock detail section showed complete and correct financial metrics:

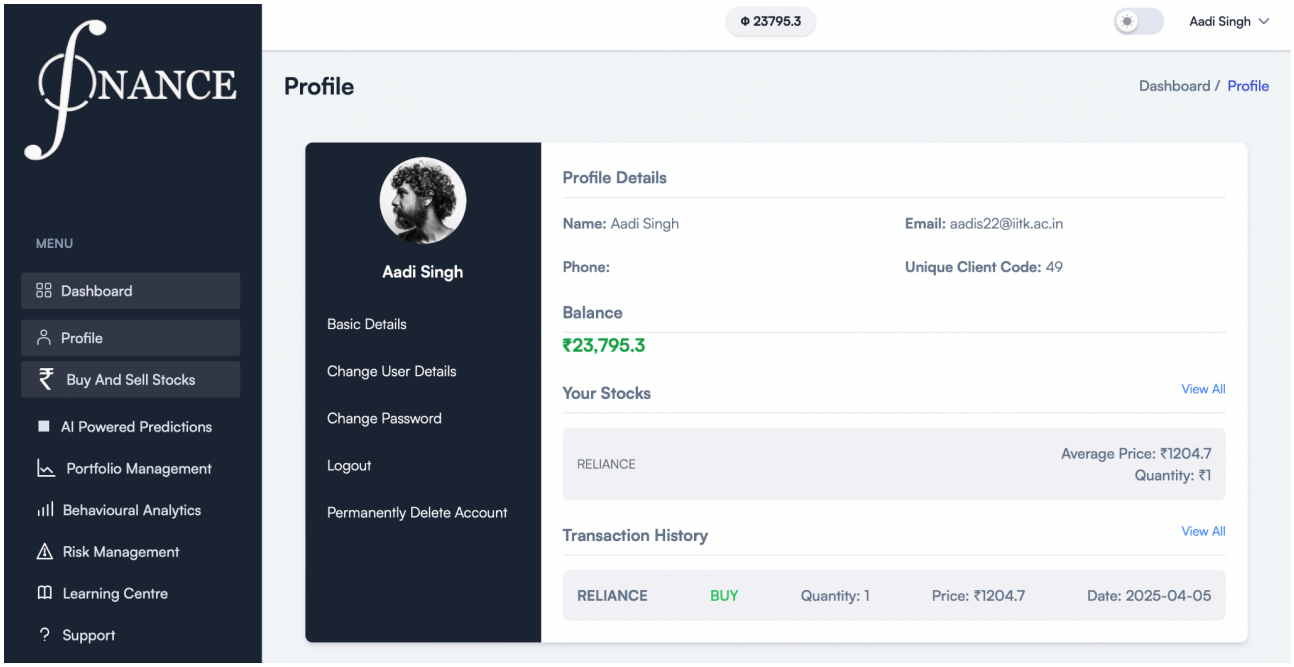
- Market Cap, Volume, EPS, P/E Ratio, Dividend Yield, Net Income, Book Value, Target Prices, etc.

No UI errors or inconsistencies were observed.

Structural Coverage:

Covered successful buy flow including balance deduction and portfolio update.

Covered rendering of extended stock metadata and chart visualization.



Unit Name: Buy Stock - RELIANCE

Unit Details: Component: Buy And Sell Stocks, Function: placeBuyOrder()

Test Owner: Sangam Gupta

Test Date: 01/04/2025

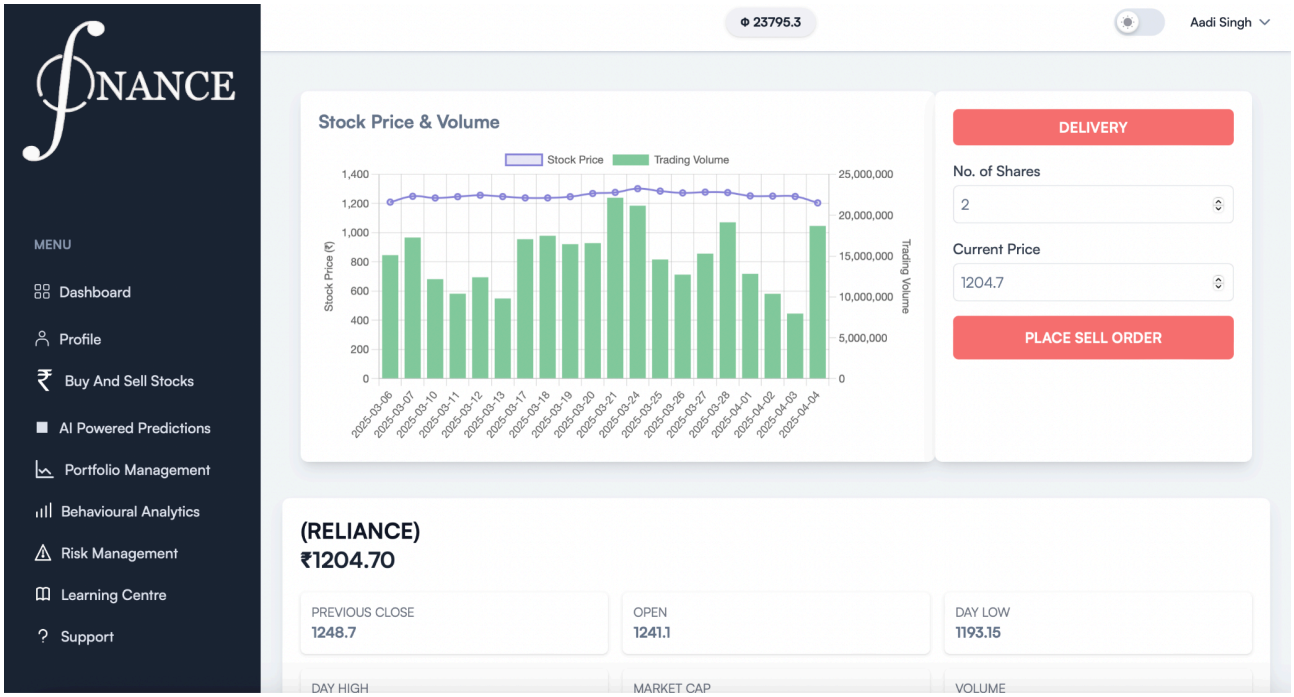
Test Results:

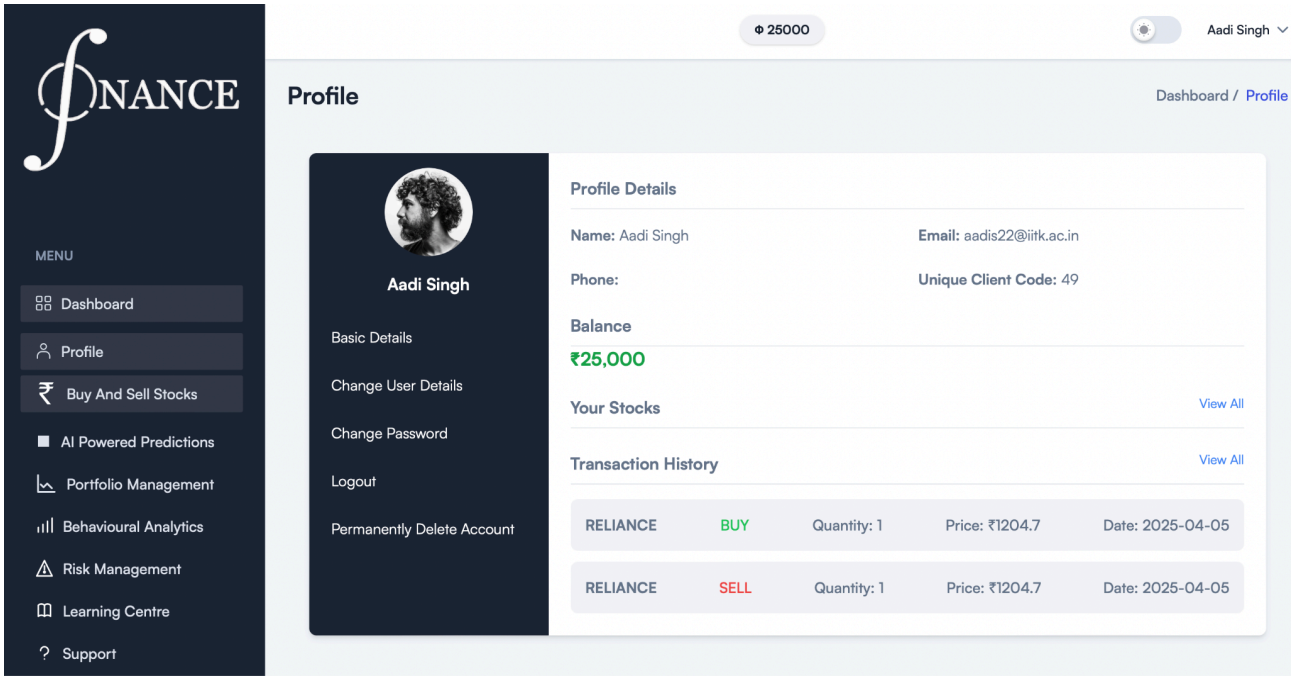
The test simulates a successful stock purchase of 1 share of RELIANCE at ₹1204.7.

Initially, the user's balance was ₹25,000. After the transaction, it reduced to ₹23,795.3, and the transaction history reflected the purchase of RELIANCE with quantity 1 at the specified price.

Structural Coverage:

- UI input validation for share quantity
- Balance update logic
- Transaction history logging
- Portfolio stock addition/update





Unit Name: Testing an invalid SELL operation when stock quantity is insufficient.

Unit Details: SellStockComponent, handleSell function

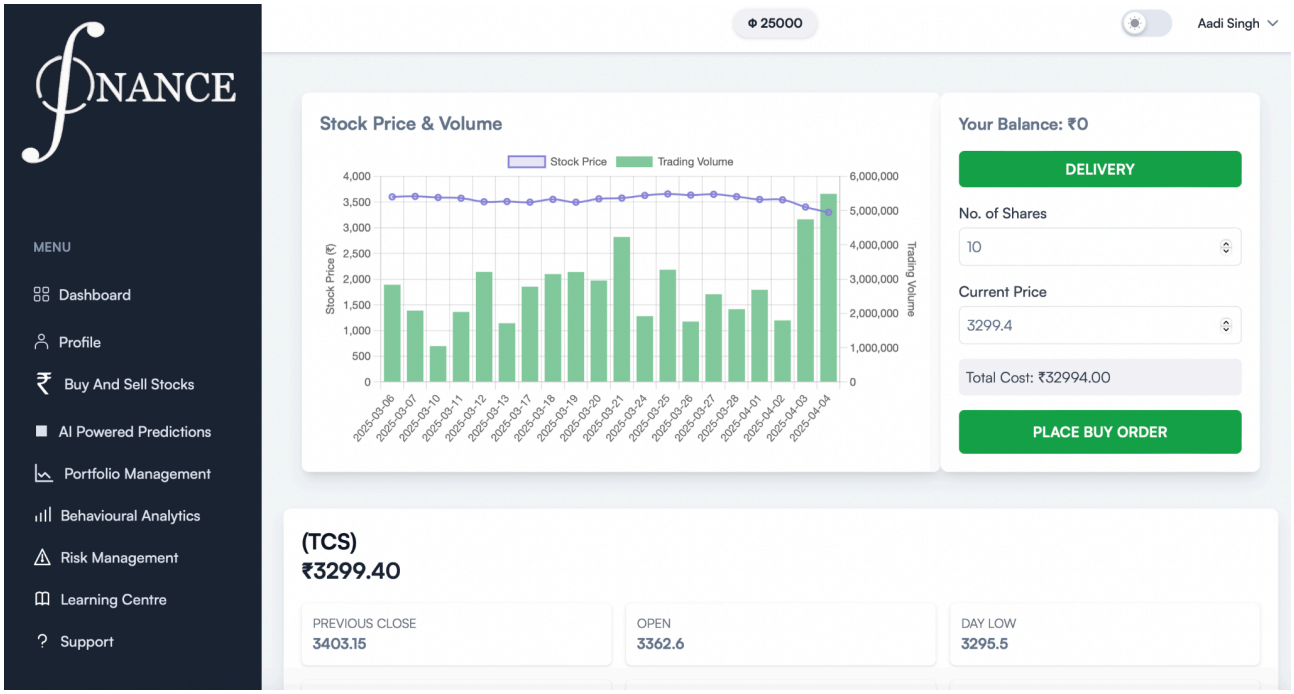
Test Owner: Aadi Singh

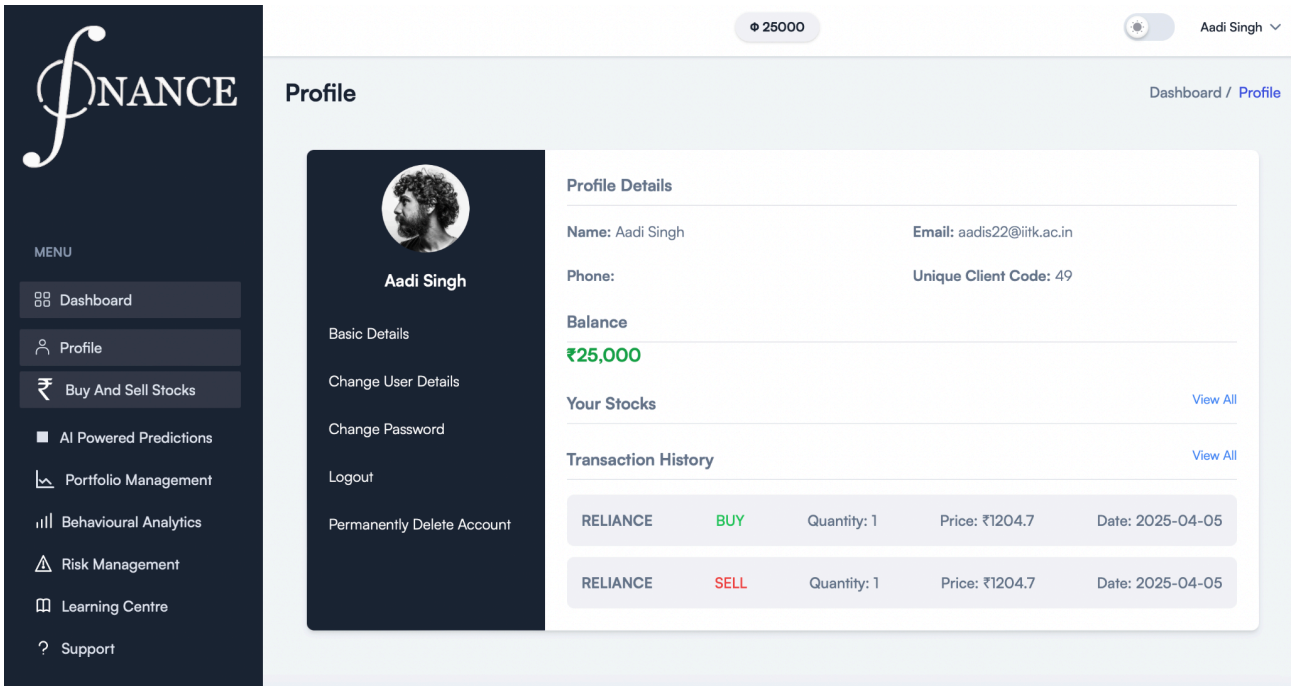
Test Date: 04/05/2025 - 04/05/2025

Test Results: User attempted to sell 2 shares of RELIANCE while holding 0 shares. No transaction was recorded in the profile, confirming the prevention of invalid operations.

Structural Coverage: Path coverage on sell function covering the 'insufficient stock' conditional branch.

Additional Comments: Function correctly blocks selling of unowned stock; no erroneous balance or transaction entry was recorded.





Unit Name: Validation of Buy Action Blocked Due to Insufficient Balance

Unit Details: BuyStockComponent, handleBuy()

Test Owner: Aadi Singh

Test Date: 02/04/2025

Test Results:

Attempted to purchase 10 shares of TCS at ₹3299.40 per share with an account balance of ₹25,000.

The action was correctly blocked as the balance was insufficient.

No updates were reflected in the profile's transaction history or stock holdings, indicating successful prevention.

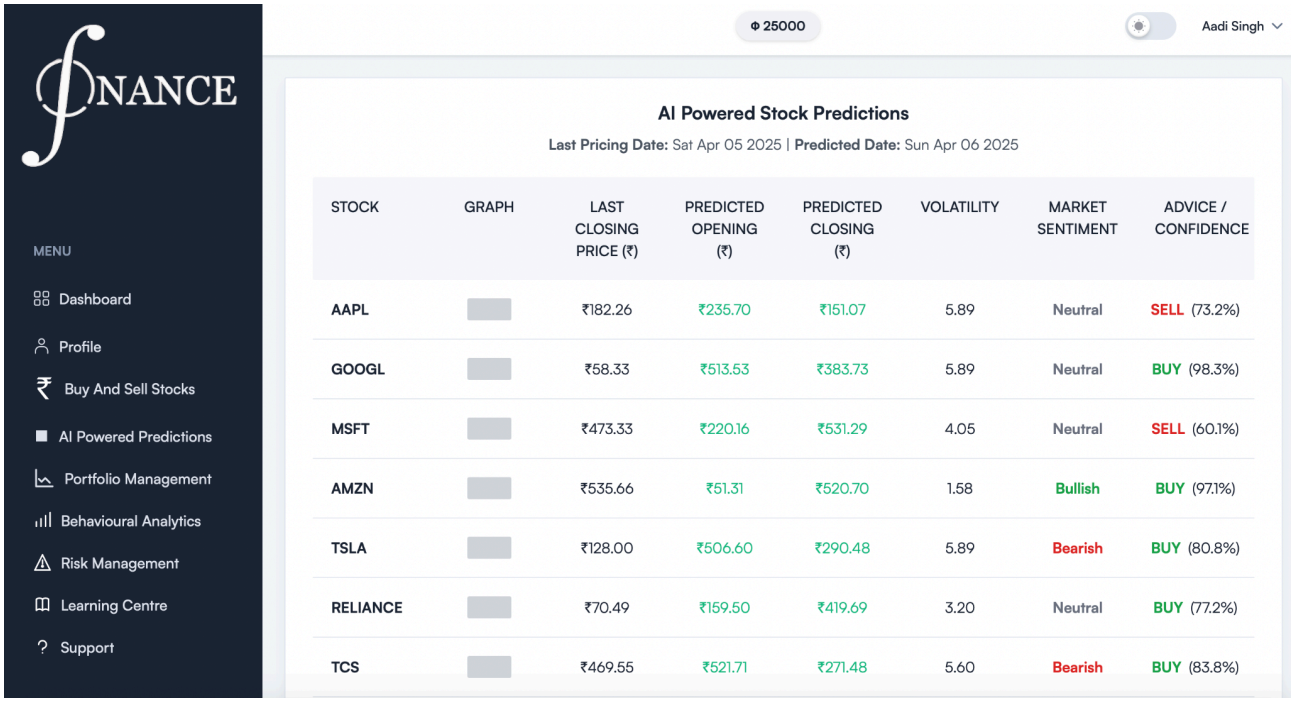
Structural Coverage:

Test case covered validation logic for buy condition, ensuring stock purchase is allowed only if sufficient funds are available.

Verified by attempting transaction and checking frontend state post-action.

Additional Comments:

The system successfully handled a negative test case.



Unit Name: AI Powered Stock Predictions – Prediction Table Validation

Unit Details:

Class: PredictionComponent

Function: fetchAndRenderPredictions()

Test Owner: Sameer Yadav

Test Date: 01/04/2025

Test Results:

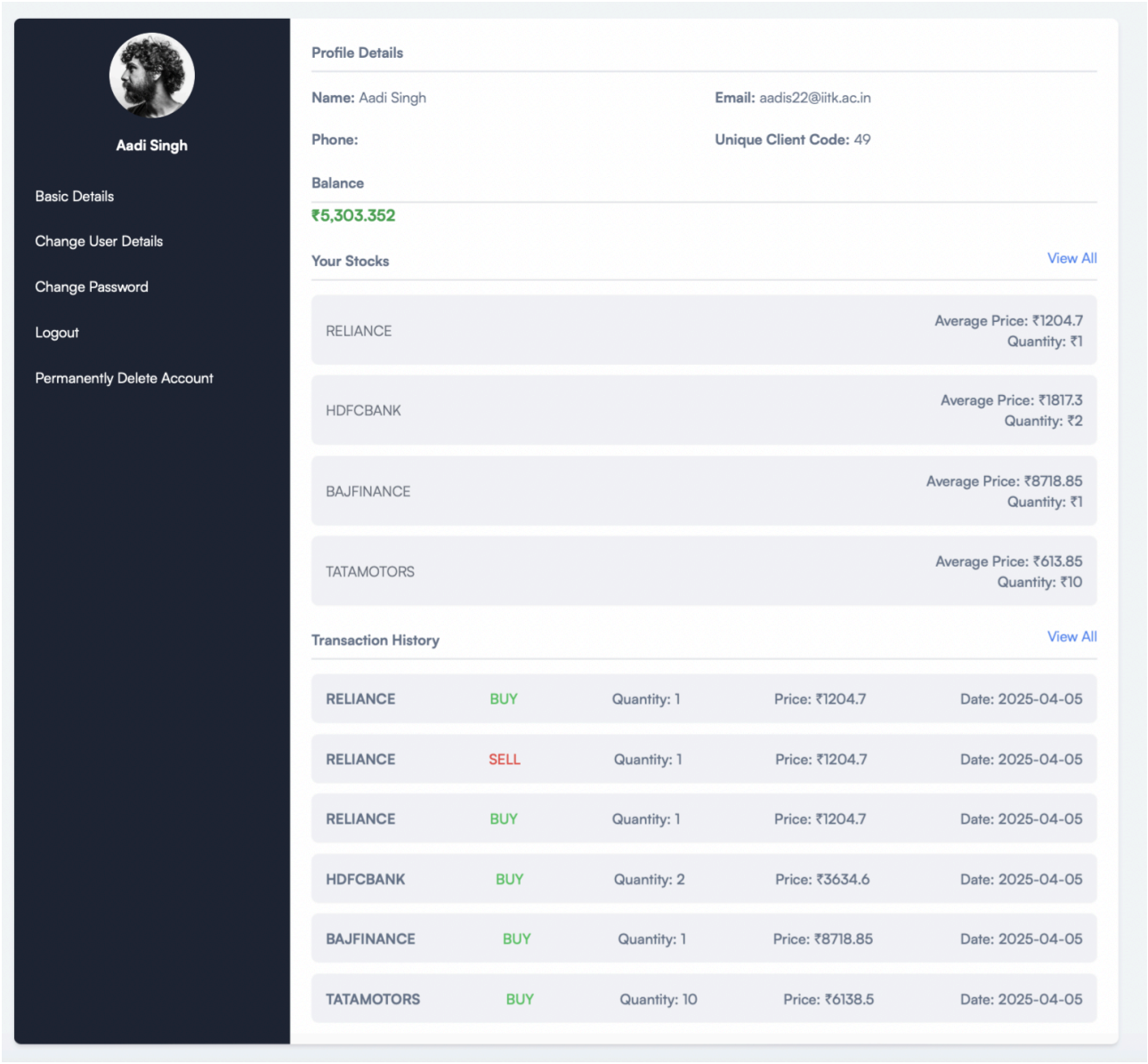
The AI-powered stock prediction module correctly displays the prediction table with the following columns:

- STOCK
- LAST CLOSING PRICE (₹)
- PREDICTED OPENING (₹)
- PREDICTED CLOSING (₹)
- VOLATILITY
- MARKET SENTIMENT
- ADVICE / CONFIDENCE

The prediction values match expected formats, sentiment and advice match the predicted changes, and confidence scores are shown in percentage. No data rendering or alignment issues observed.

Structural Coverage:

Test case includes a visual check of the table content rendering after fetching backend data. Coverage includes validation for correct price formatting, sentiment labeling, and confidence value display. Also confirmed correct visual segregation using colors for BUY/SELL tags.



Unit Name: Profile Overview and Transaction History Module

Unit Details: ProfilePage.tsx, TransactionHistory.tsx – Handles display of user profile details, stock holdings, and transaction logs.

Test Owner: Aadya Dhir

Test Date: 04/04/2025

Test Results:

- The profile section correctly displays the user's name, email, balance, and unique client

code.

- All user-owned stocks (RELIANCE, HDFCBANK, BAJFINANCE, TATAMOTORS) are listed with correct average prices and quantities.

- The transaction history accurately logs all buy/sell activities with ticker, quantity, price, and date details.

- Transactions include:

  • 2 BUY and 1 SELL for RELIANCE

  • 1 BUY for HDFCBANK (2 units)

  • 1 BUY for BAJFINANCE

  • 1 BUY for TATAMOTORS (10 units)

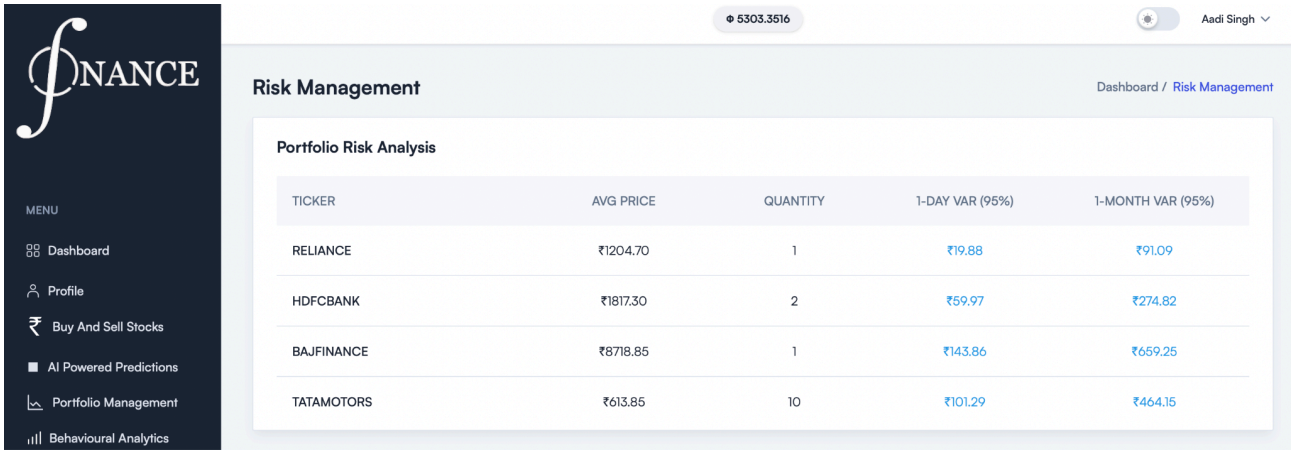
Structural Coverage:

- Path Coverage: All critical rendering paths on the profile page were verified.

- Conditional Coverage: Validated conditions for different transaction types (BUY/SELL) and dynamic content rendering.

Additional Comments:

The transaction list supports both BUY and SELL types with accurate formatting and alignment. No functional or visual bugs were found. The displayed balance reflects post-transaction state.



Unit Name: Risk Management - Portfolio Risk Analysis View Functionality

Unit Details: RiskManagementComponent, fetchRiskData()

Test Owner: Aruz Awasthi

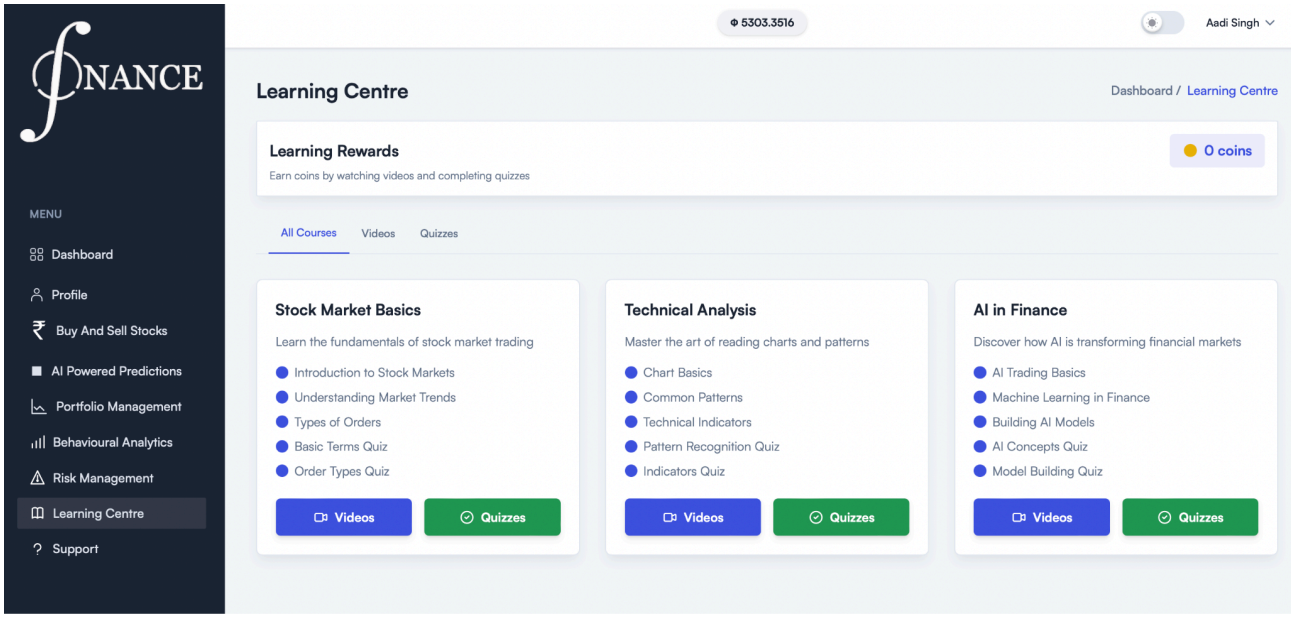
Test Date: 02/04/2025

Test Results:

The test verified the rendering of the Portfolio Risk Analysis component, including accurate display of tickers, average prices, quantities, and calculated 1-day and 1-month Value at Risk (VaR) at 95% confidence level. All data fields populated correctly for four stocks (RELIANCE, HDFCBANK, BAJFINANCE, TATAMOTORS). The interface functioned as expected with no errors.

Structural Coverage:

Branch coverage was achieved for conditional rendering based on available risk data. The data mapping and formatting logic for currency and percentage values was exercised. Coverage included both the component's initial data load and rendering states.



Unit Name: Learning Centre Page - Display and Functionality Validation

Unit Details:

Class: LearningCentre.jsx

Function: Renders educational content including videos and quizzes under three course categories: Stock Market Basics, Technical Analysis, and AI in Finance.

Test Owner: Dilbar Singh Lamba

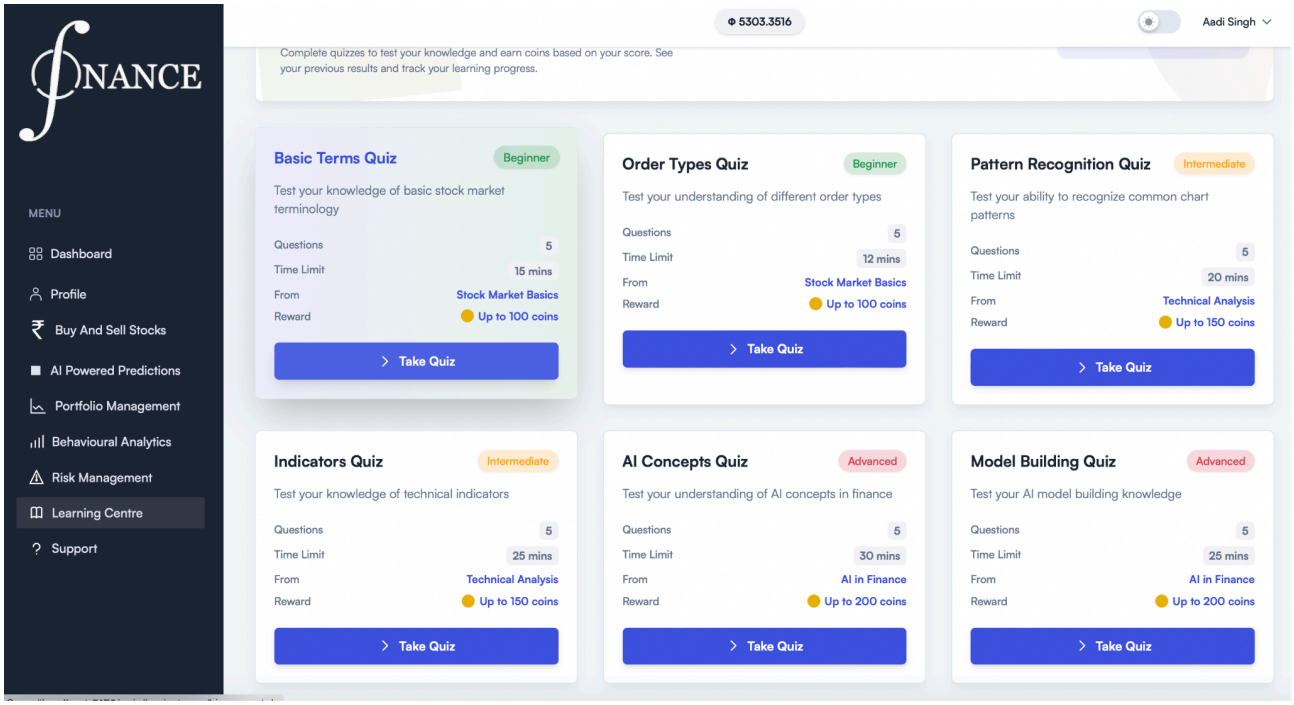
Test Date: 03/04/2025

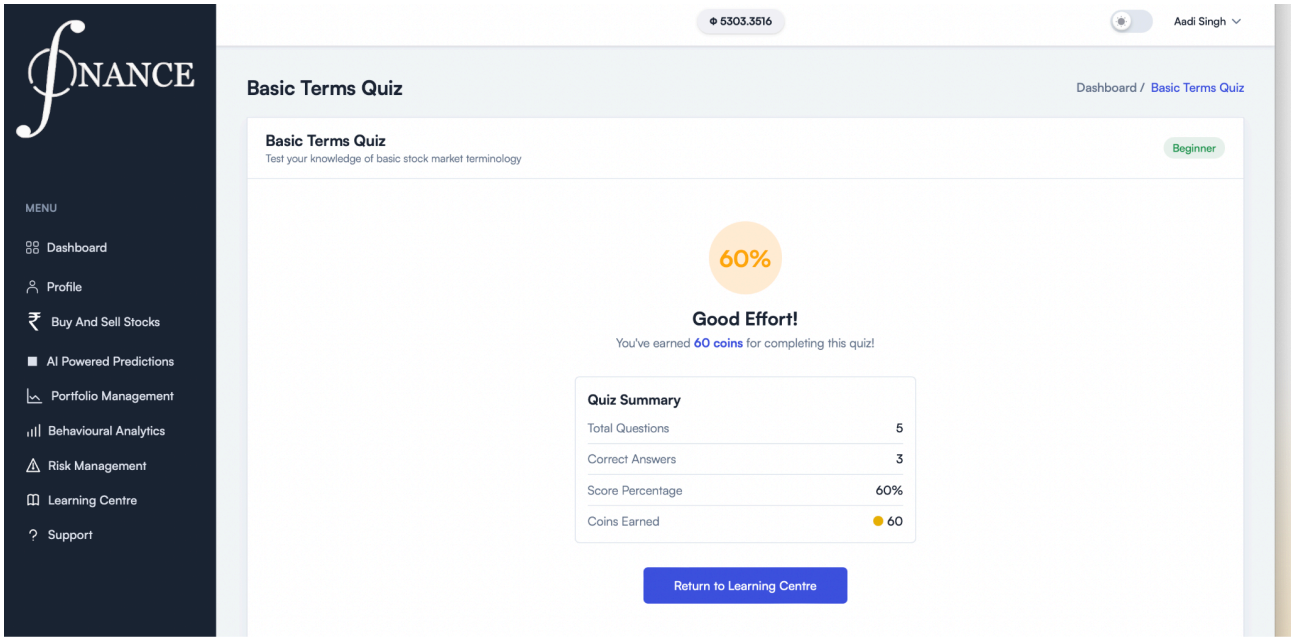
Test Results:

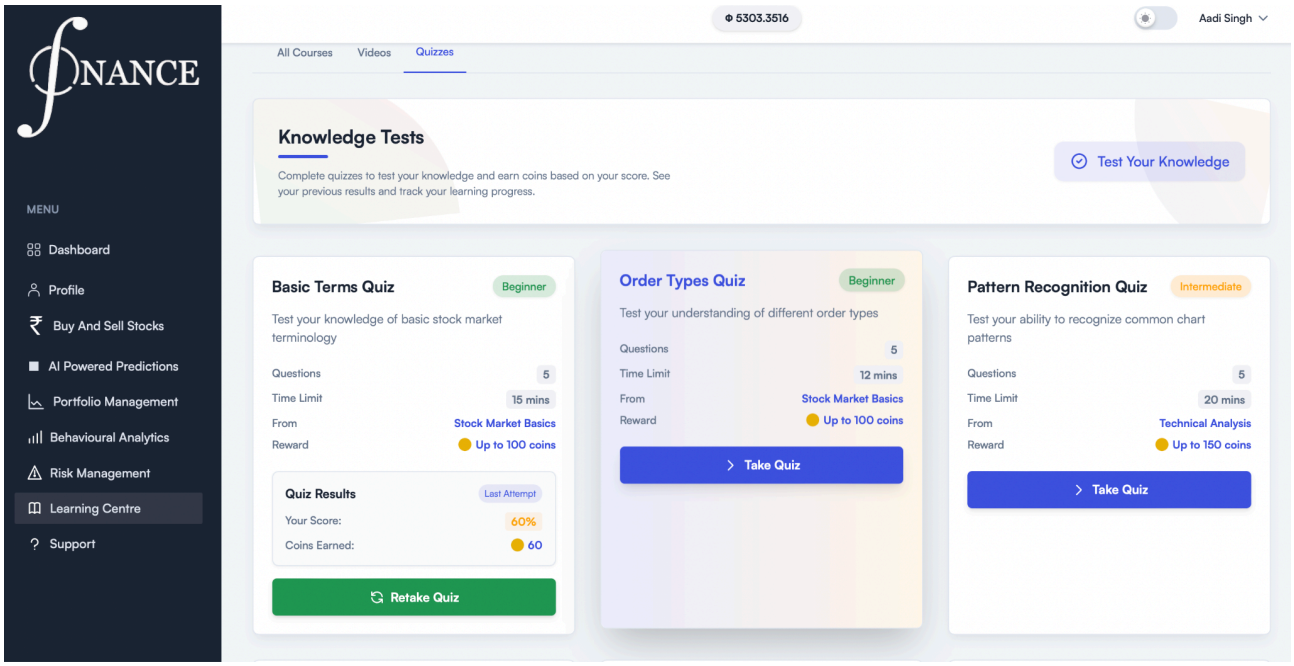
The test verified that all course categories and their respective modules were displayed correctly. The "Videos" and "Quizzes" buttons were visible and clickable for each category. Navigation tabs (All Courses, Videos, Quizzes) were functioning as expected. Coin balance display and "Learning Rewards" header were present. All text elements and sections rendered without distortion or layout issues.

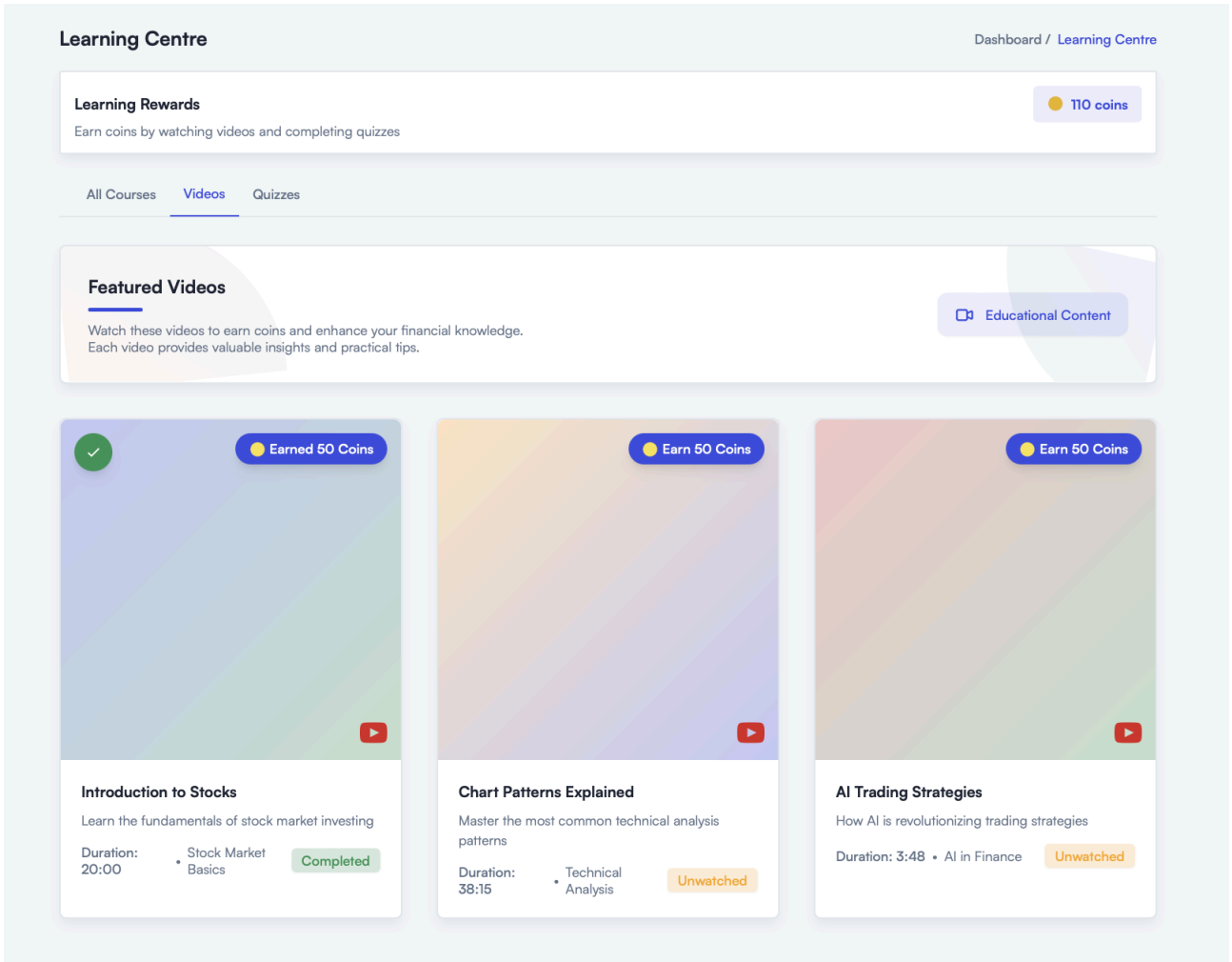
Structural Coverage:

Tested through UI component rendering validation and interaction simulation using Cypress. Covered component rendering logic, button presence, and basic routing behavior on click events.


NANCE
5303.3516
Aadi Singh
Complete quizzes to test your knowledge and earn coins based on your score. See your previous results and track your learning progress.
MENU
Dashboard
Profile
Buy And Sell Stocks
AI Powered Predictions
Portfolio Management
Behavioural Analytics
Risk Management
Learning Centre
Support
Basic Terms Quiz
Beginner
Test your knowledge of basic stock market terminology
Questions 5
Time Limit 15 mins
From Stock Market Basics
Reward Up to 100 coins
Take Quiz
Order Types Quiz
Beginner
Test your understanding of different order types
Questions 5
Time Limit 12 mins
From Stock Market Basics
Reward Up to 100 coins
Take Quiz
Pattern Recognition Quiz
Intermediate
Test your ability to recognize common chart patterns
Questions 5
Time Limit 20 mins
From Technical Analysis
Reward Up to 150 coins
Take Quiz
Indicators Quiz
Intermediate
Test your knowledge of technical indicators
Questions 5
Time Limit 25 mins
From Technical Analysis
Reward Up to 150 coins
Take Quiz
AI Concepts Quiz
Advanced
Test your understanding of AI concepts in finance
Questions 5
Time Limit 30 mins
From AI in Finance
Reward Up to 200 coins
Take Quiz
Model Building Quiz
Advanced
Test your AI model building knowledge
Questions 5
Time Limit 25 mins
From AI in Finance
Reward Up to 200 coins
Take Quiz


NANCE
5303.3516
Aadi Singh
Basic Terms Quiz
Dashboard / Basic Terms Quiz
MENU
Dashboard
Profile
Buy And Sell Stocks
AI Powered Predictions
Portfolio Management
Behavioural Analytics
Risk Management
Learning Centre
Support
Basic Terms Quiz
Test your knowledge of basic stock market terminology
Beginner
60%
Good Effort!
You've earned 60 coins for completing this quiz!
Quiz Summary
Total Questions 5
Correct Answers 3
Score Percentage 60%
Coins Earned 60
Return to Learning Centre


FINANCE
MENU
Dashboard
Profile
Buy And Sell Stocks
AI Powered Predictions
Portfolio Management
Behavioural Analytics
Risk Management
Learning Centre
Support
5303.3516
Aadi Singh
All Courses
Videos
Quizzes
Knowledge Tests
Complete quizzes to test your knowledge and earn coins based on your score. See your previous results and track your learning progress.
Test Your Knowledge
Basic Terms Quiz
Beginner
Test your knowledge of basic stock market terminology
Questions 5
Time Limit 15 mins
From Stock Market Basics
Reward Up to 100 coins
Quiz Results
Last Attempt
Your Score: 60%
Coins Earned: 60
Retake Quiz
Order Types Quiz
Beginner
Test your understanding of different order types
Questions 5
Time Limit 12 mins
From Stock Market Basics
Reward Up to 100 coins
Take Quiz
Pattern Recognition Quiz
Intermediate
Test your ability to recognize common chart patterns
Questions 5
Time Limit 20 mins
From Technical Analysis
Reward Up to 150 coins
Take Quiz


Learning Centre
Dashboard / Learning Centre
Learning Rewards
Earn coins by watching videos and completing quizzes
110 coins
All Courses
Videos
Quizzes
Featured Videos
Watch these videos to earn coins and enhance your financial knowledge. Each video provides valuable insights and practical tips.
Educational Content
Earned 50 Coins
Introduction to Stocks
Learn the fundamentals of stock market investing
Duration: 20:00
Stock Market Basics
Completed
Earn 50 Coins
Chart Patterns Explained
Master the most common technical analysis patterns
Duration: 38:15
Technical Analysis
Unwatched
Earn 50 Coins
AI Trading Strategies
How AI is revolutionizing trading strategies
Duration: 3:48
AI in Finance
Unwatched

Unit Name: Learning Centre – Quizzes and Videos Rewards Functionality

[Tests the reward and progress tracking system after taking quizzes and watching educational videos.]

Unit Details:

Component: LearningCentre.tsx

Functions: handleVideoWatch, submitQuiz, calculateCoinReward

Test Owner: Anisha Srivastava

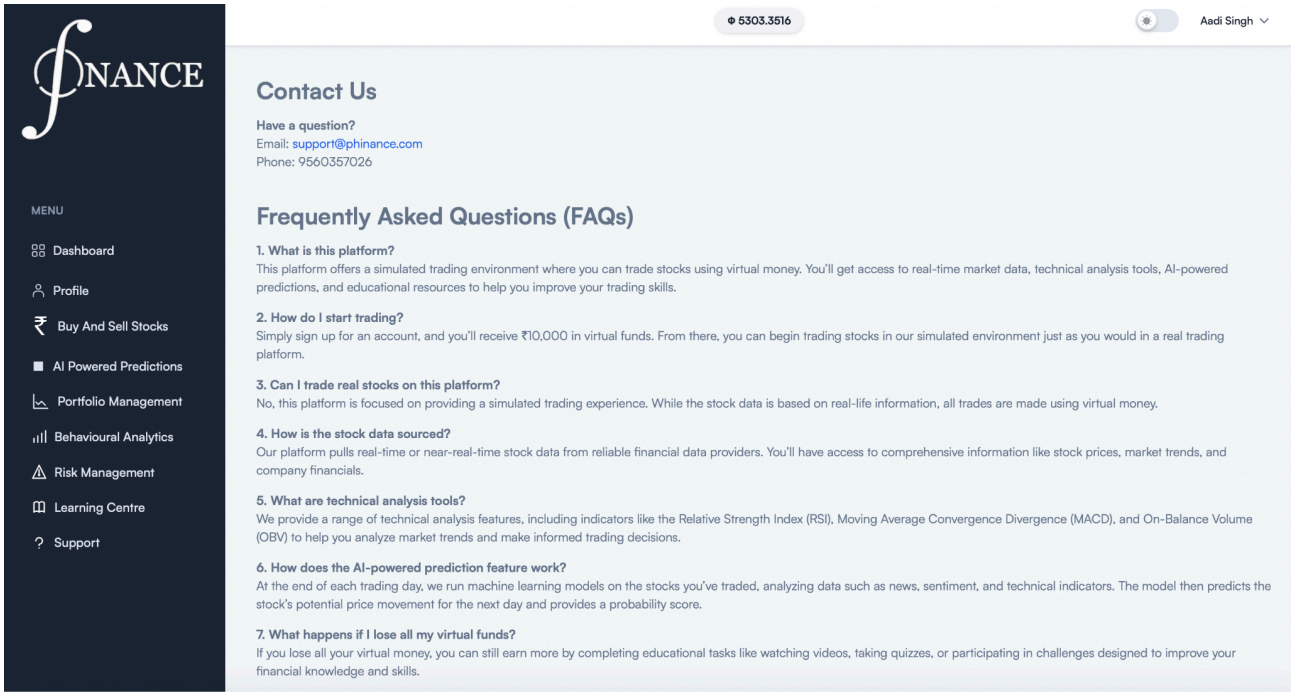
Test Date: 01/04/2025

Test Results:

- User watched the “Introduction to Stocks” video and was awarded 50 coins successfully.
- User completed the “Basic Terms Quiz” with a 60% score, receiving 60 coins.
- “Order Types Quiz” and “Pattern Recognition Quiz” are visible but not attempted yet.
- Coin balance updated correctly to 110 coins.
- Badge “Earned 50 Coins” correctly displayed on watched content.

Structural Coverage:

- Conditional rendering based on user action (quiz taken, video watched).
- Reward allocation logic executed successfully.
- State updates for coin balance verified.



Unit Name: FAQ Page - Contact Info and Dynamic FAQ Section Rendering

Unit Details:

Component: ContactUsPage.tsx

Functions: renderFAQ(), renderContactInfo()

Test Owner: Pranshu Thirani

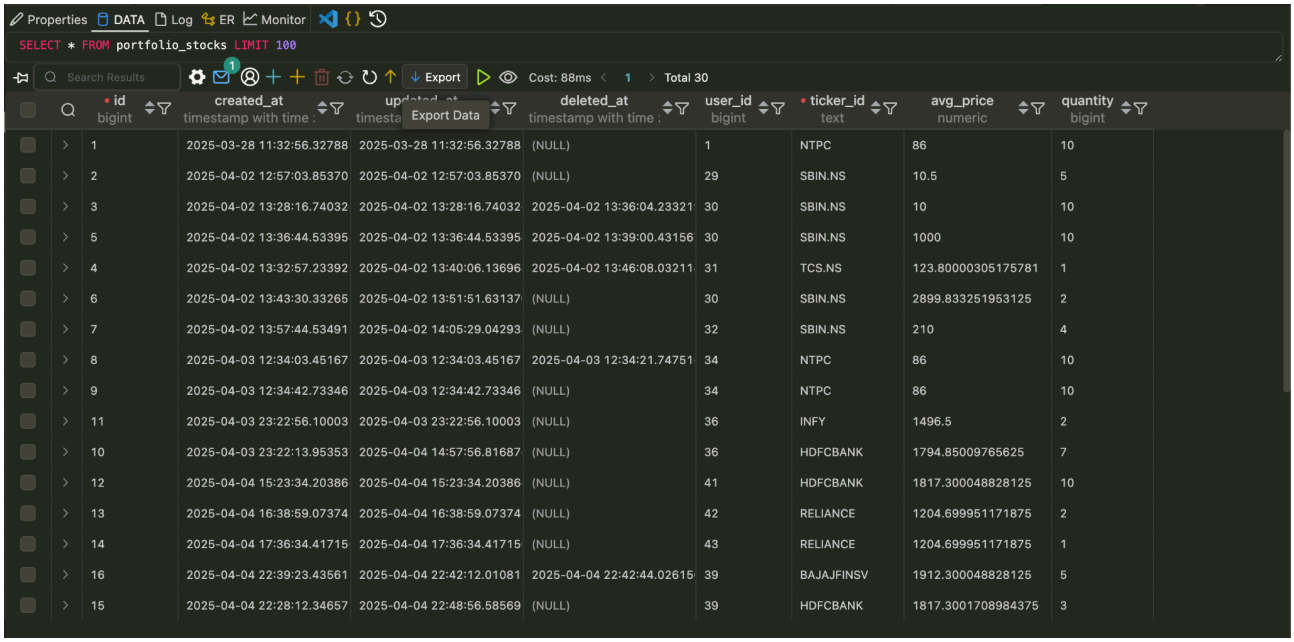
Test Date: 31/03/2025

Test Results:

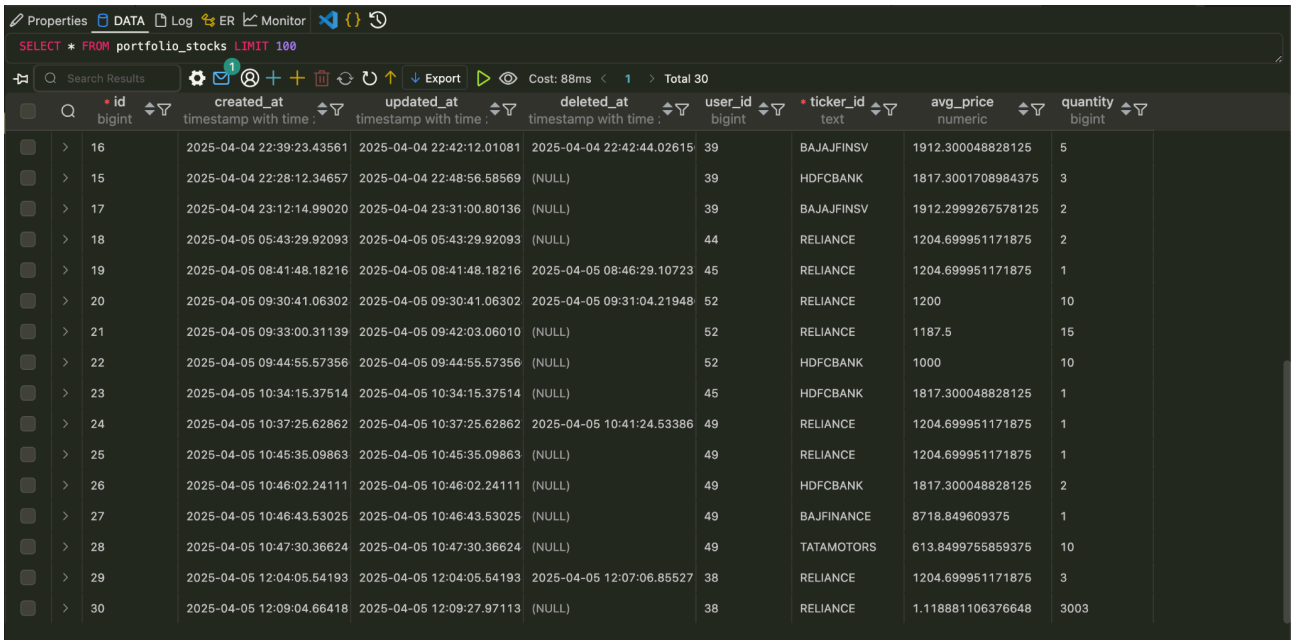
- Verified correct rendering of contact email and phone.
- Verified FAQ entries are fully and correctly displayed.
- No UI overlap or rendering issues detected.

Structural Coverage:

- Manual visual inspection and UI behavior check
- Static content validation
- Edge case check: no unexpected layout break on resize



id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	avg_price numeric	quantity bigint
1	2025-03-28 11:32:56.32788	2025-03-28 11:32:56.32788	(NULL)	1	NTPC	86	10
2	2025-04-02 12:57:03.85370	2025-04-02 12:57:03.85370	(NULL)	29	SBIN.NS	10.5	5
3	2025-04-02 13:28:16.74032	2025-04-02 13:28:16.74032	2025-04-02 13:36:04.23321	30	SBIN.NS	10	10
5	2025-04-02 13:36:44.53395	2025-04-02 13:36:44.53395	2025-04-02 13:39:00.43156	30	SBIN.NS	1000	10
4	2025-04-02 13:32:57.23392	2025-04-02 13:40:06.13696	2025-04-02 13:46:08.03211	31	TCS.NS	123.80000305175781	1
6	2025-04-02 13:43:30.33265	2025-04-02 13:51:51.63137	(NULL)	30	SBIN.NS	2899.833251953125	2
7	2025-04-02 13:57:44.53491	2025-04-02 14:05:29.04293	(NULL)	32	SBIN.NS	210	4
8	2025-04-03 12:34:03.45167	2025-04-03 12:34:03.45167	2025-04-03 12:34:21.74751	34	NTPC	86	10
9	2025-04-03 12:34:42.73346	2025-04-03 12:34:42.73346	(NULL)	34	NTPC	86	10
11	2025-04-03 23:22:56.10003	2025-04-03 23:22:56.10003	(NULL)	36	INFY	1496.5	2
10	2025-04-03 23:22:13.95353	2025-04-04 14:57:56.81687	(NULL)	36	HDFCBANK	1794.85009765625	7
12	2025-04-04 15:23:34.20386	2025-04-04 15:23:34.20386	(NULL)	41	HDFCBANK	1817.300048828125	10
13	2025-04-04 16:38:59.07374	2025-04-04 16:38:59.07374	(NULL)	42	RELIANCE	1204.699951171875	2
14	2025-04-04 17:36:34.41715	2025-04-04 17:36:34.41715	(NULL)	43	RELIANCE	1204.699951171875	1
16	2025-04-04 22:39:23.43561	2025-04-04 22:42:12.01081	2025-04-04 22:42:44.02615	39	BAJAJFINSV	1912.300048828125	5
15	2025-04-04 22:28:12.34657	2025-04-04 22:48:56.58569	(NULL)	39	HDFCBANK	1817.3001708984375	3



id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	avg_price numeric	quantity bigint
16	2025-04-04 22:39:23.43561	2025-04-04 22:42:12.01081	2025-04-04 22:42:44.02615	39	BAJAJFINSV	1912.300048828125	5
15	2025-04-04 22:28:12.34657	2025-04-04 22:48:56.58569	(NULL)	39	HDFCBANK	1817.3001708984375	3
17	2025-04-04 23:12:14.99020	2025-04-04 23:31:00.80136	(NULL)	39	BAJAJFINSV	1912.2999267578125	2
18	2025-04-05 05:43:29.92093	2025-04-05 05:43:29.92093	(NULL)	44	RELIANCE	1204.699951171875	2
19	2025-04-05 08:41:48.18216	2025-04-05 08:41:48.18216	2025-04-05 08:46:29.10723	45	RELIANCE	1204.699951171875	1
20	2025-04-05 09:30:41.06302	2025-04-05 09:30:41.06302	2025-04-05 09:31:04.21948	52	RELIANCE	1200	10
21	2025-04-05 09:33:00.31139	2025-04-05 09:42:03.06010	(NULL)	52	RELIANCE	1187.5	15
22	2025-04-05 09:44:55.57356	2025-04-05 09:44:55.57356	(NULL)	52	HDFCBANK	1000	10
23	2025-04-05 10:34:15.37514	2025-04-05 10:34:15.37514	(NULL)	45	HDFCBANK	1817.300048828125	1
24	2025-04-05 10:37:25.62862	2025-04-05 10:37:25.62862	2025-04-05 10:41:24.53386	49	RELIANCE	1204.699951171875	1
25	2025-04-05 10:45:35.09863	2025-04-05 10:45:35.09863	(NULL)	49	RELIANCE	1204.699951171875	1
26	2025-04-05 10:46:02.24111	2025-04-05 10:46:02.24111	(NULL)	49	HDFCBANK	1817.300048828125	2
27	2025-04-05 10:46:43.53025	2025-04-05 10:46:43.53025	(NULL)	49	BAJFINANCE	8718.849609375	1
28	2025-04-05 10:47:30.36624	2025-04-05 10:47:30.36624	(NULL)	49	TATAMOTORS	613.8499755859375	10
29	2025-04-05 12:04:05.54193	2025-04-05 12:04:05.54193	2025-04-05 12:07:06.85527	38	RELIANCE	1204.699951171875	3
30	2025-04-05 12:09:04.66418	2025-04-05 12:09:27.97113	(NULL)	38	RELIANCE	1.118881106376648	3003

Unit Name: Portfolio Stocks Database Testing

This unit test verifies that the portfolio stocks database is functioning as expected. The test ensures that transactions submitted via the Postman client and the frontend queries are correctly updating and reflecting in the database.

Unit Details:

Module: Portfolio Stocks Database

The Portfolio_stocks database has the following columns:

- id - index of the database
- created_at - the timestamp at which the database entry was created
- updated_at - the timestamp at which the database entry was updated
- deleted_at - the timestamp at which the database entry was deleted
- user_id - user id of the user
- ticker_id - ticker id of the stock owned in the portfolio
- avg_price - the price at which the user bought the stocks
- quantity - quantity of the stock owned

Functions Tested: Data submission via Postman and frontend queries, transaction logging, and data retrieval

Test Owner: Anish Sahu

Test Date: 04/04/2025

Test Results:

- Transactions initiated through the Postman client were successfully processed and became visible in the database.

- Queries submitted from the frontend accurately retrieved the expected data.

- The database consistently reflected all transaction updates, ensuring that the query results matched the previously made transactions.

Structural Coverage:

- The testing focused on verifying the endpoints used for data submission and retrieval through the Postman client and frontend interface.

- Each query route was exercised to ensure that the correct transaction data is logged and displayed.

- Structural coverage includes validation of data flow from the client to the database and back to the user interface.

Properties DATA Log ER Monitor

```
SELECT * FROM tickers LIMIT 100
```

Search Results · Export · Cost: 89ms · 1 · Total 39

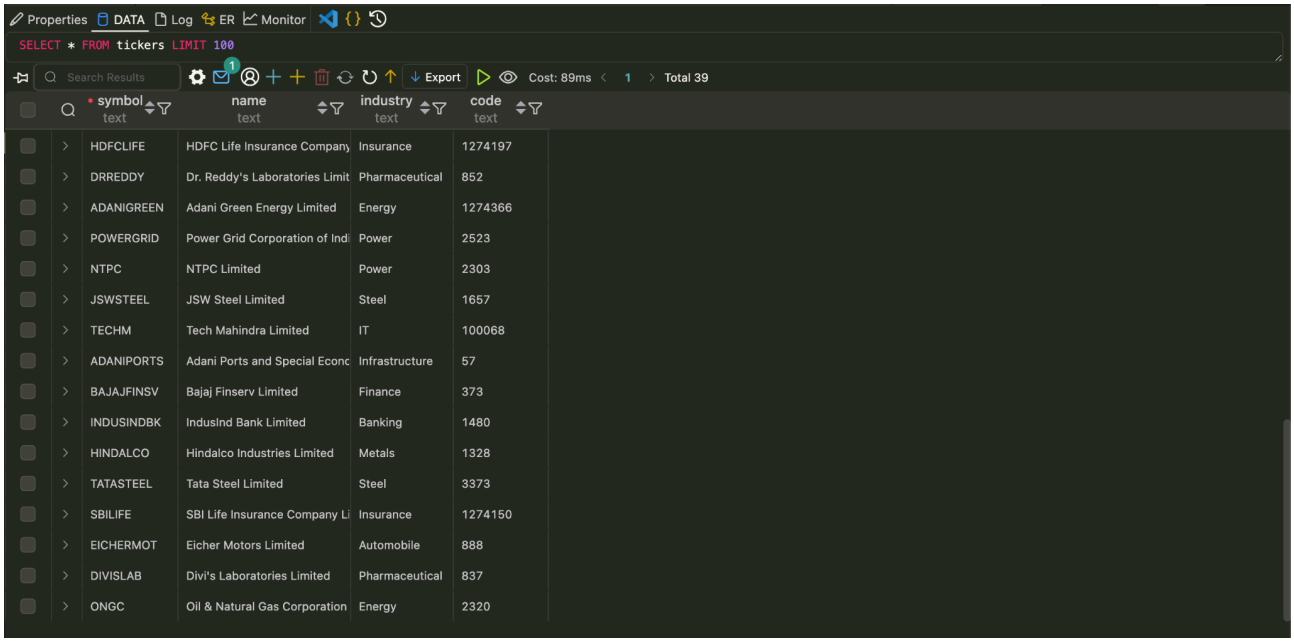
* symbol text	name text	industry text	code text
RELIANCE	Reliance Industries Limited	Energy	2726
HDFCBANK	HDFC Bank Limited	Banking	1298
TCS	Tata Consultancy Services Lim	IT	3365
BHARTIARTL	Bharti Airtel Limited	Telecom	467
ICICIBANK	ICICI Bank Limited	Banking	1384
SBIN	State Bank of India	Banking	3188
INFY	Infosys Limited	IT	1489
BAJFINANCE	Bajaj Finance Limited	Finance	372
HINDUNILVR	Hindustan Unilever Limited	FMCG	1350
ITC	ITC Limited	FMCG	1552
KOTAKBANK	Kotak Mahindra Bank Limited	Banking	1818
LT	Larsen & Toubro Limited	Infrastructure	1870
HCLTECH	HCL Technologies Limited	IT	1297
MARUTI	Maruti Suzuki India Limited	Automobile	2023
SUNPHARMA	Sun Pharmaceutical Industries	Pharmaceutical	3245
ASIANPAINT	Asian Paints Limited	Paints	295

Properties DATA Log ER Monitor

```
SELECT * FROM tickers LIMIT 100
```

Search Results · Export · Cost: 89ms · 1 · Total 39

* symbol text	name text	industry text	code text
AXISBANK	Axis Bank Limited	Banking	348
TITAN	Titan Company Limited	Consumer Goods	3437
ULTRACEMCO	UltraTech Cement Limited	Cement	3525
WIPRO	Wipro Limited	IT	3763
NESTLEIND	Nestlé India Limited	FMCG	2236
M&M	Mahindra & Mahindra Limited	Automobile	1973
TATAMOTORS	Tata Motors Limited	Automobile	3370
HDFCLIFE	HDFC Life Insurance Company	Insurance	1274197
DRREDDY	Dr. Reddy's Laboratories Limit	Pharmaceutical	852
ADANIGREEN	Adani Green Energy Limited	Energy	1274366
POWERGRID	Power Grid Corporation of Indi	Power	2523
NTPC	NTPC Limited	Power	2303
JSWSTEEL	JSW Steel Limited	Steel	1657
TECHM	Tech Mahindra Limited	IT	100068
ADANIPORTS	Adani Ports and Special Econc	Infrastructure	57
BAJAJFINSV	Bajaj Finserv Limited	Finance	373



Unit Name: Tickers Database Testing
This unit test verifies that the tickers database correctly maintains and displays the list of all stock tickers.

Unit Details:
Module: Tickers Database

The Tickers Database has 4 columns:

- symbol - the ticker symbol of the stock
- name - name of the stock
- industry - industry where the company is based
- code - unique code number of the stock

Functions Tested: Data submission and retrieval through the Postman client and frontend queries, ensuring that the complete list of tickers is accurately stored and displayed.

Test Owner: Aruz Awasthi

Test Date: 04/04/2025

Test Results:

- The tickers database holds the list of all stocks used by us in our website, namely 50 stocks of the nifty50 list.

- The complete list of stock tickers was accurately retrieved and displayed in the dashboard, confirming that no entries were omitted.
- The system consistently reflected all modifications, verifying that the data flow from input to storage was functioning correctly.

Structural Coverage:

- Testing focused on the endpoints handling data input and retrieval for the tickers database.
- The test ensured that the list of tickers remains comprehensive and up-to-date through multiple submissions and queries.

Properties DATA Log ER Monitor

```
SELECT * FROM transactions LIMIT 100
```

Search Results | Export | Cost: 93ms | 1 | Total 74

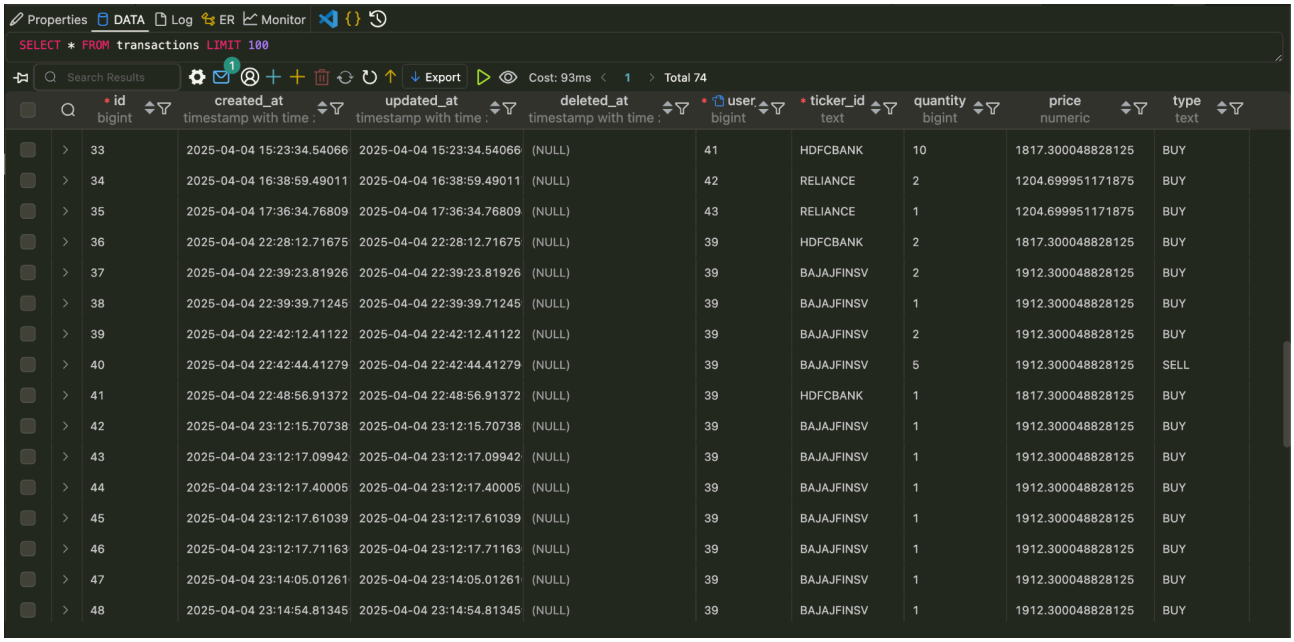
id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	quantity bigint	price numeric	type text
1	2025-03-28 11:32:56.84946	2025-03-28 11:32:56.84946	(NULL)	1	NTPC	10	86	BUY
2	2025-04-02 12:57:03.86919	2025-04-02 12:57:03.86919	(NULL)	29	SBIN.NS	5	10.5	BUY
3	2025-04-02 13:28:16.75075	2025-04-02 13:28:16.75075	(NULL)	30	SBIN.NS	10	10	BUY
4	2025-04-02 13:32:57.23973	2025-04-02 13:32:57.23973	(NULL)	31	TCS.NS	4	123.80000305175781	BUY
5	2025-04-02 13:34:05.54037	2025-04-02 13:34:05.54037	(NULL)	31	TCS.NS	4	123.80000305175781	BUY
6	2025-04-02 13:34:34.04285	2025-04-02 13:34:34.04285	(NULL)	31	TCS.NS	3	100.5	SELL
7	2025-04-02 13:36:04.24123	2025-04-02 13:36:04.24123	(NULL)	30	SBIN.NS	10	100	SELL
8	2025-04-02 13:36:44.5401+	2025-04-02 13:36:44.5401+	(NULL)	30	SBIN.NS	10	1000	BUY
9	2025-04-02 13:38:55.04114	2025-04-02 13:38:55.04114	(NULL)	31	TCS.NS	3	1000	SELL
10	2025-04-02 13:39:00.43791	2025-04-02 13:39:00.43791	(NULL)	30	SBIN.NS	10	1000	SELL
11	2025-04-02 13:40:06.14345	2025-04-02 13:40:06.14345	(NULL)	31	TCS.NS	1	1000	SELL
12	2025-04-02 13:43:30.33868	2025-04-02 13:43:30.33868	(NULL)	30	SBIN.NS	1	2900	BUY
13	2025-04-02 13:43:52.34260	2025-04-02 13:43:52.34260	(NULL)	30	SBIN.NS	1	2900	BUY
14	2025-04-02 13:44:11.5392+	2025-04-02 13:44:11.5392+	(NULL)	30	SBIN.NS	1	2900	BUY
15	2025-04-02 13:45:34.94188	2025-04-02 13:45:34.94188	(NULL)	30	SBIN.NS	1	2900	BUY
16	2025-04-02 13:46:08.03875	2025-04-02 13:46:08.03875	(NULL)	31	TCS.NS	1	100.5	SELL

Properties DATA Log ER Monitor

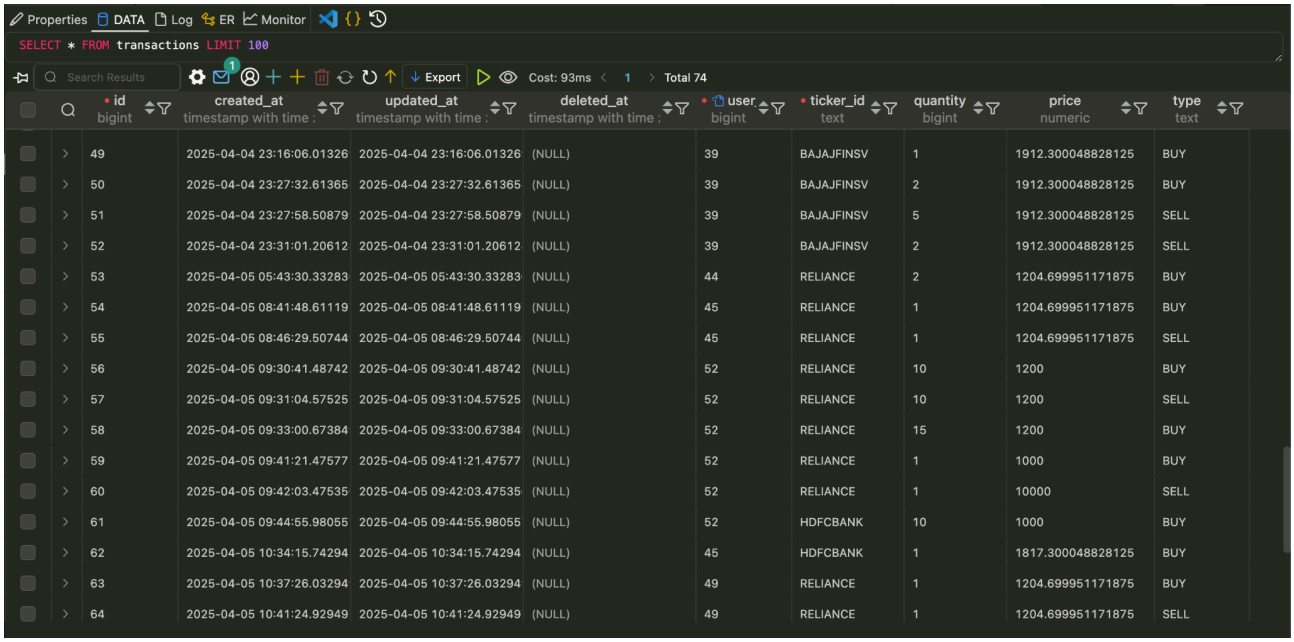
```
SELECT * FROM transactions LIMIT 100
```

Search Results | Export | Cost: 93ms | 1 | Total 74

id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	quantity bigint	price numeric	type text
17	2025-04-02 13:46:14.63906	2025-04-02 13:46:14.63906	(NULL)	30	SBIN.NS	1	2900	BUY
18	2025-04-02 13:50:30.04283	2025-04-02 13:50:30.04283	(NULL)	30	SBIN.NS	1	2899	BUY
19	2025-04-02 13:51:18.63952	2025-04-02 13:51:18.63952	(NULL)	30	SBIN.NS	1	1	SELL
20	2025-04-02 13:51:38.84002	2025-04-02 13:51:38.84002	(NULL)	30	SBIN.NS	1	1	SELL
21	2025-04-02 13:51:40.13966	2025-04-02 13:51:40.13966	(NULL)	30	SBIN.NS	1	1	SELL
22	2025-04-02 13:51:51.63747	2025-04-02 13:51:51.63747	(NULL)	30	SBIN.NS	1	0.5	SELL
23	2025-04-02 13:57:44.54126	2025-04-02 13:57:44.54126	(NULL)	32	SBIN.NS	4	210	BUY
24	2025-04-02 13:58:38.73817	2025-04-02 13:58:38.73817	(NULL)	32	SBIN.NS	4	210	BUY
25	2025-04-02 14:05:29.04921	2025-04-02 14:05:29.04921	(NULL)	32	SBIN.NS	4	210	SELL
26	2025-04-03 12:34:04.36178	2025-04-03 12:34:04.36178	(NULL)	34	NTPC	10	86	BUY
27	2025-04-03 12:34:22.53268	2025-04-03 12:34:22.53268	(NULL)	34	NTPC	10	86	SELL
28	2025-04-03 12:34:43.42666	2025-04-03 12:34:43.42666	(NULL)	34	NTPC	10	86	BUY
29	2025-04-03 23:22:14.34480	2025-04-03 23:22:14.34480	(NULL)	36	HDFCBANK	4	1794.8499755859375	BUY
30	2025-04-03 23:22:56.51662	2025-04-03 23:22:56.51662	(NULL)	36	INFY	2	1496.5	BUY
31	2025-04-03 23:27:03.32057	2025-04-03 23:27:03.32057	(NULL)	36	HDFCBANK	5	1794.8499755859375	BUY
32	2025-04-04 14:57:57.23308	2025-04-04 14:57:57.23308	(NULL)	36	HDFCBANK	2	1817.300048828125	SELL



id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_ bigint	ticker_id text	quantity bigint	price numeric	type text
33	2025-04-04 15:23:34.54066	2025-04-04 15:23:34.54066	(NULL)	41	HDFCBANK	10	1817.300048828125	BUY
34	2025-04-04 16:38:59.49011	2025-04-04 16:38:59.49011	(NULL)	42	RELIANCE	2	1204.699951171875	BUY
35	2025-04-04 17:36:34.76809	2025-04-04 17:36:34.76809	(NULL)	43	RELIANCE	1	1204.699951171875	BUY
36	2025-04-04 22:28:12.71675	2025-04-04 22:28:12.71675	(NULL)	39	HDFCBANK	2	1817.300048828125	BUY
37	2025-04-04 22:39:23.81926	2025-04-04 22:39:23.81926	(NULL)	39	BAJAJFINSV	2	1912.300048828125	BUY
38	2025-04-04 22:39:39.71245	2025-04-04 22:39:39.71245	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
39	2025-04-04 22:42:12.41122	2025-04-04 22:42:12.41122	(NULL)	39	BAJAJFINSV	2	1912.300048828125	BUY
40	2025-04-04 22:42:44.41279	2025-04-04 22:42:44.41279	(NULL)	39	BAJAJFINSV	5	1912.300048828125	SELL
41	2025-04-04 22:48:56.91372	2025-04-04 22:48:56.91372	(NULL)	39	HDFCBANK	1	1817.300048828125	BUY
42	2025-04-04 23:12:15.70738	2025-04-04 23:12:15.70738	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
43	2025-04-04 23:12:17.09942	2025-04-04 23:12:17.09942	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
44	2025-04-04 23:12:17.40005	2025-04-04 23:12:17.40005	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
45	2025-04-04 23:12:17.61039	2025-04-04 23:12:17.61039	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
46	2025-04-04 23:12:17.71163	2025-04-04 23:12:17.71163	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
47	2025-04-04 23:14:05.01261	2025-04-04 23:14:05.01261	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
48	2025-04-04 23:14:54.81345	2025-04-04 23:14:54.81345	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY

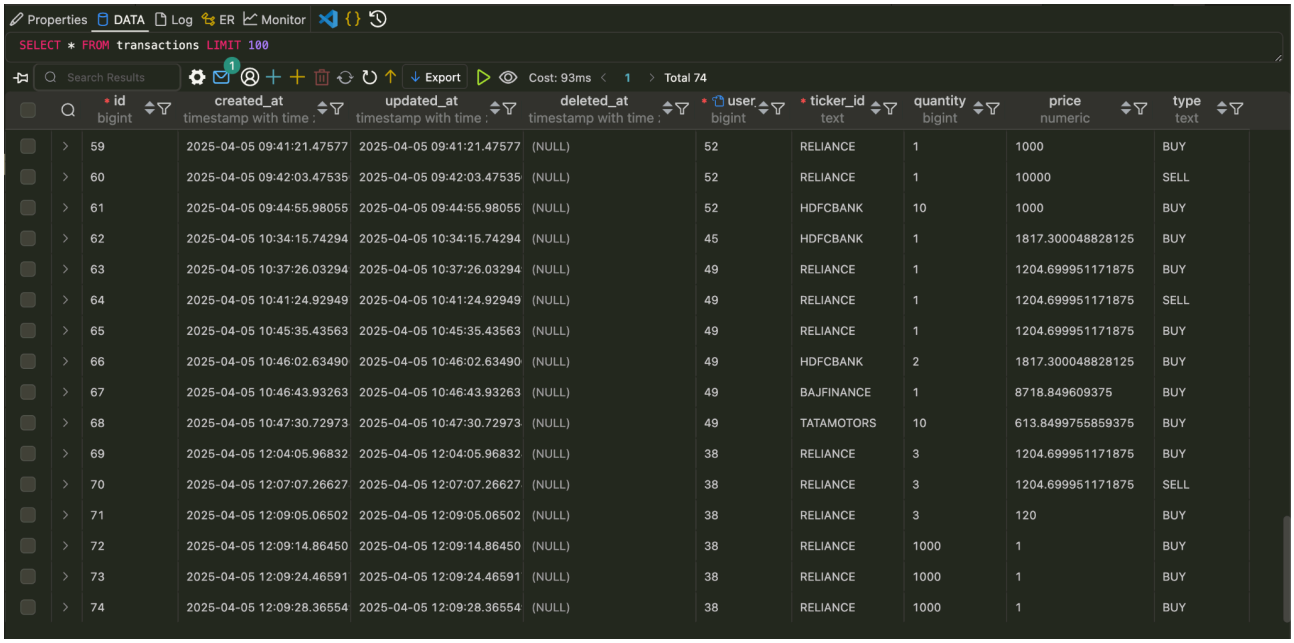


Properties DATA Log ER Monitor

```
SELECT * FROM transactions LIMIT 100
```

Search Results | Export | Cost: 93ms | 1 | Total 74

id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	quantity bigint	price numeric	type text
49	2025-04-04 23:16:06.01326	2025-04-04 23:16:06.01326	(NULL)	39	BAJAJFINSV	1	1912.300048828125	BUY
50	2025-04-04 23:27:32.61365	2025-04-04 23:27:32.61365	(NULL)	39	BAJAJFINSV	2	1912.300048828125	BUY
51	2025-04-04 23:27:58.50879	2025-04-04 23:27:58.50879	(NULL)	39	BAJAJFINSV	5	1912.300048828125	SELL
52	2025-04-04 23:31:01.20612	2025-04-04 23:31:01.20612	(NULL)	39	BAJAJFINSV	2	1912.300048828125	SELL
53	2025-04-05 05:43:30.33283	2025-04-05 05:43:30.33283	(NULL)	44	RELIANCE	2	1204.699951171875	BUY
54	2025-04-05 08:41:48.61119	2025-04-05 08:41:48.61119	(NULL)	45	RELIANCE	1	1204.699951171875	BUY
55	2025-04-05 08:46:29.50744	2025-04-05 08:46:29.50744	(NULL)	45	RELIANCE	1	1204.699951171875	SELL
56	2025-04-05 09:30:41.48742	2025-04-05 09:30:41.48742	(NULL)	52	RELIANCE	10	1200	BUY
57	2025-04-05 09:31:04.57525	2025-04-05 09:31:04.57525	(NULL)	52	RELIANCE	10	1200	SELL
58	2025-04-05 09:33:00.67384	2025-04-05 09:33:00.67384	(NULL)	52	RELIANCE	15	1200	BUY
59	2025-04-05 09:41:21.47577	2025-04-05 09:41:21.47577	(NULL)	52	RELIANCE	1	1000	BUY
60	2025-04-05 09:42:03.47535	2025-04-05 09:42:03.47535	(NULL)	52	RELIANCE	1	10000	SELL
61	2025-04-05 09:44:55.98055	2025-04-05 09:44:55.98055	(NULL)	52	HDFCBANK	10	1000	BUY
62	2025-04-05 10:34:15.74294	2025-04-05 10:34:15.74294	(NULL)	45	HDFCBANK	1	1817.300048828125	BUY
63	2025-04-05 10:37:26.03294	2025-04-05 10:37:26.03294	(NULL)	49	RELIANCE	1	1204.699951171875	BUY
64	2025-04-05 10:41:24.92949	2025-04-05 10:41:24.92949	(NULL)	49	RELIANCE	1	1204.699951171875	SELL



Properties DATA Log ER Monitor

```
SELECT * FROM transactions LIMIT 100
```

Search Results | Export | Cost: 93ms | 1 | Total 74

id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	user_id bigint	ticker_id text	quantity bigint	price numeric	type text
59	2025-04-05 09:41:21.47577	2025-04-05 09:41:21.47577	(NULL)	52	RELIANCE	1	1000	BUY
60	2025-04-05 09:42:03.47535	2025-04-05 09:42:03.47535	(NULL)	52	RELIANCE	1	10000	SELL
61	2025-04-05 09:44:55.98055	2025-04-05 09:44:55.98055	(NULL)	52	HDFCBANK	10	1000	BUY
62	2025-04-05 10:34:15.74294	2025-04-05 10:34:15.74294	(NULL)	45	HDFCBANK	1	1817.300048828125	BUY
63	2025-04-05 10:37:26.03294	2025-04-05 10:37:26.03294	(NULL)	49	RELIANCE	1	1204.699951171875	BUY
64	2025-04-05 10:41:24.92949	2025-04-05 10:41:24.92949	(NULL)	49	RELIANCE	1	1204.699951171875	SELL
65	2025-04-05 10:45:35.43563	2025-04-05 10:45:35.43563	(NULL)	49	RELIANCE	1	1204.699951171875	BUY
66	2025-04-05 10:46:02.63490	2025-04-05 10:46:02.63490	(NULL)	49	HDFCBANK	2	1817.300048828125	BUY
67	2025-04-05 10:46:43.93263	2025-04-05 10:46:43.93263	(NULL)	49	BAJFINANCE	1	8718.849609375	BUY
68	2025-04-05 10:47:30.72973	2025-04-05 10:47:30.72973	(NULL)	49	TATAMOTORS	10	613.8499755859375	BUY
69	2025-04-05 12:04:05.96832	2025-04-05 12:04:05.96832	(NULL)	38	RELIANCE	3	1204.699951171875	BUY
70	2025-04-05 12:07:07.26627	2025-04-05 12:07:07.26627	(NULL)	38	RELIANCE	3	1204.699951171875	SELL
71	2025-04-05 12:09:05.06502	2025-04-05 12:09:05.06502	(NULL)	38	RELIANCE	3	120	BUY
72	2025-04-05 12:09:14.86450	2025-04-05 12:09:14.86450	(NULL)	38	RELIANCE	1000	1	BUY
73	2025-04-05 12:09:24.46591	2025-04-05 12:09:24.46591	(NULL)	38	RELIANCE	1000	1	BUY
74	2025-04-05 12:09:28.36554	2025-04-05 12:09:28.36554	(NULL)	38	RELIANCE	1000	1	BUY

Unit Name: Transactions Database Testing

This unit test verifies that the transactions database is functioning as expected. The test ensures that transactions submitted via the Postman client and the frontend queries are correctly updating and reflecting in the database.

Unit Details:

Module: Transactions Database

The Transactions database has the following columns:

- id - index of the database
- created_at - the timestamp at which the database entry was created
- updated_at - the timestamp at which the database entry was updated
- deleted_at - the timestamp at which the database entry was deleted
- user_id - user id of the user that completed the transaction
- ticker_id- the symbols of the stocks that were bought/sold.
- quantity - quantity of the stock bought/sold.
- price- avg price of the stock in the transaction
- type- type of transaction: Buy or Sell

Functions Tested: Data submission via Postman and frontend queries, transaction logging, and data retrieval

Test Owner: Sameer Yadav

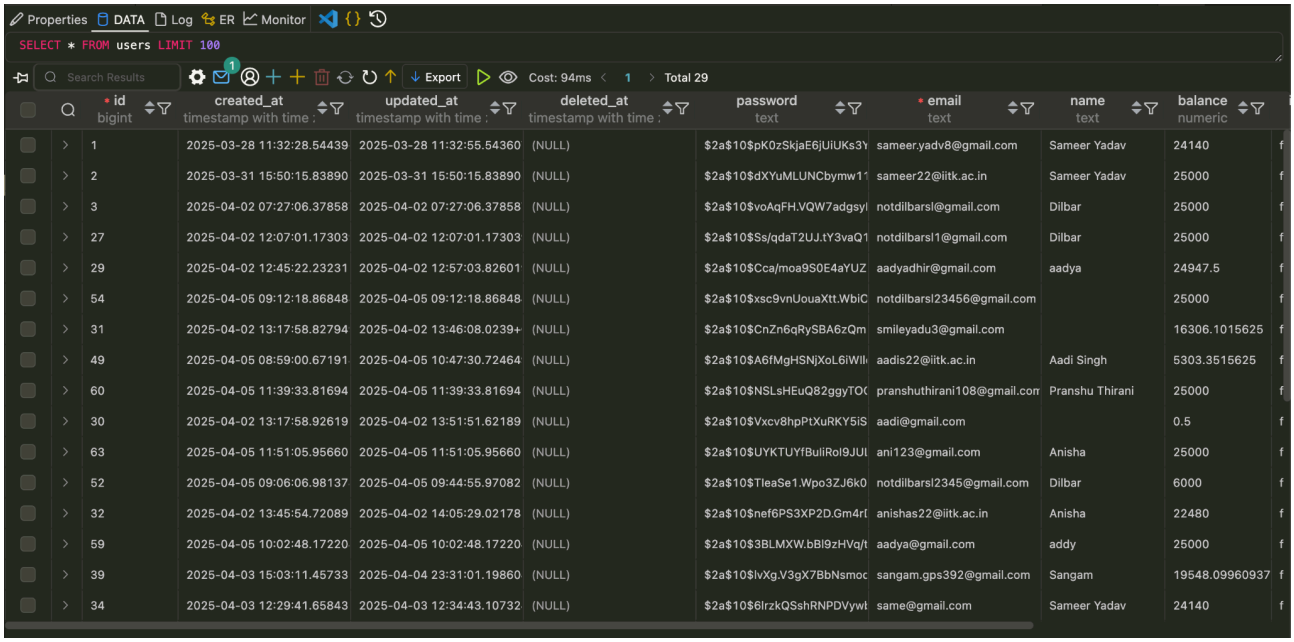
Test Date: 04/04/2025

Test Results:

- Transactions initiated through the Postman client were successfully processed and became visible in the database
- Queries submitted from the frontend accurately retrieved the expected data.
- The database consistently reflected all transaction updates, ensuring that the query results matched the previously made transactions.

Structural Coverage:

- The testing focused on verifying the endpoints used for data submission and retrieval through the Postman client and frontend interface.
- Each query route was exercised to ensure that the correct transaction data is logged and displayed.
- Structural coverage includes validation of data flow from the client to the database and back to the user interface.

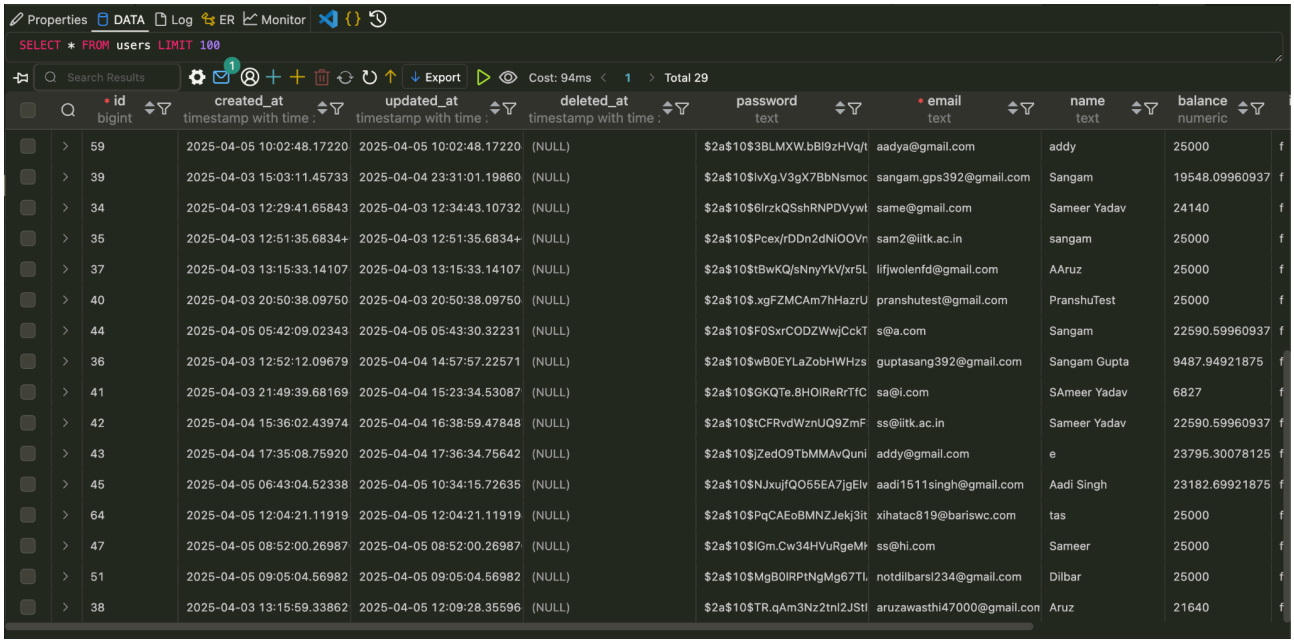


Properties DATA Log ER Monitor

```
SELECT * FROM users LIMIT 100
```

Search Results · Export · Cost: 94ms · 1 · Total 29

id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	password text	email text	name text	balance numeric	
1	2025-03-28 11:32:28.54439	2025-03-28 11:32:55.54360	(NULL)	$2a$10$pK0zSkjaE6jUiUKs3Y	sameer.yadv8@gmail.com	Sameer Yadav	24140	f
2	2025-03-31 15:50:15.83890	2025-03-31 15:50:15.83890	(NULL)	$2a$10$dXYuMLUNCbymw11	sameer22@iitk.ac.in	Sameer Yadav	25000	f
3	2025-04-02 07:27:06.37858	2025-04-02 07:27:06.37858	(NULL)	$2a$10$voAqFH.VQW7adgsyl	notdilbarsl@gmail.com	Dilbar	25000	f
27	2025-04-02 12:07:01.17303	2025-04-02 12:07:01.17303	(NULL)	$2a$10$Ss/qdaT2UJ.tY3vaQ1	notdilbarsl1@gmail.com	Dilbar	25000	f
29	2025-04-02 12:45:22.23231	2025-04-02 12:57:03.82601	(NULL)	$2a$10$Cca/moa9S0E4aYUZ	aadyadhir@gmail.com	aadya	24947.5	f
54	2025-04-05 09:12:18.86848	2025-04-05 09:12:18.86848	(NULL)	$2a$10$xsc9vnUouaXtt.WbiC	notdilbarsl23456@gmail.com		25000	f
31	2025-04-02 13:17:58.82794	2025-04-02 13:46:08.0239+	(NULL)	$2a$10$CnZn6qRySBA6zQm	smileyadu3@gmail.com		16306.1015625	f
49	2025-04-05 08:59:00.67191	2025-04-05 10:47:30.72464	(NULL)	$2a$10$A6fMgHSNjXoL6iWIl	aadis22@iitk.ac.in	Aadi Singh	5303.3515625	f
60	2025-04-05 11:39:33.81694	2025-04-05 11:39:33.81694	(NULL)	$2a$10$NSLsHEuQ82ggyTOC	pranshuthirani108@gmail.com	Pranshu Thirani	25000	f
30	2025-04-02 13:17:58.92619	2025-04-02 13:51:51.62189	(NULL)	$2a$10$Vxcv8hpPtXuRKY5iS	aadi@gmail.com		0.5	f
63	2025-04-05 11:51:05.95660	2025-04-05 11:51:05.95660	(NULL)	$2a$10$UYKTUYfBuliRol9JUl	ani123@gmail.com	Anisha	25000	f
52	2025-04-05 09:06:06.98137	2025-04-05 09:44:55.97082	(NULL)	$2a$10$TleaSe1.Wpo3ZJ6k0	notdilbarsl2345@gmail.com	Dilbar	6000	f
32	2025-04-02 13:45:54.72089	2025-04-02 14:05:29.02178	(NULL)	$2a$10$nef6PS3XP2D.Gm4rD	anishas22@iitk.ac.in	Anisha	22480	f
59	2025-04-05 10:02:48.17220	2025-04-05 10:02:48.17220	(NULL)	$2a$10$3BLMXW.bBl9zHVq/t	aadya@gmail.com	addy	25000	f
39	2025-04-03 15:03:11.45733	2025-04-04 23:31:01.19860	(NULL)	$2a$10$lvXg.V3gX7BbNsmoc	sangam.gps392@gmail.com	Sangam	19548.09960937	f
34	2025-04-03 12:29:41.65843	2025-04-03 12:34:43.10732	(NULL)	$2a$10$6IrzkQSshRNPDVywk	same@gmail.com	Sameer Yadav	24140	f



Properties DATA Log ER Monitor

```
SELECT * FROM users LIMIT 100
```

Search Results · Export · Cost: 94ms · 1 · Total 29

id bigint	created_at timestamp with time	updated_at timestamp with time	deleted_at timestamp with time	password text	email text	name text	balance numeric	
59	2025-04-05 10:02:48.17220	2025-04-05 10:02:48.17220	(NULL)	$2a$10$3BLMXW.bBl9zHVq/t	aadya@gmail.com	addy	25000	f
39	2025-04-03 15:03:11.45733	2025-04-04 23:31:01.19860	(NULL)	$2a$10$lvXg.V3gX7BbNsmoc	sangam.gps392@gmail.com	Sangam	19548.09960937	f
34	2025-04-03 12:29:41.65843	2025-04-03 12:34:43.10732	(NULL)	$2a$10$6IrzkQSshRNPDVywk	same@gmail.com	Sameer Yadav	24140	f
35	2025-04-03 12:51:35.6834+	2025-04-03 12:51:35.6834+	(NULL)	$2a$10$Pcex/rDDn2dNiOOVn	sam2@iitk.ac.in	sangam	25000	f
37	2025-04-03 13:15:33.14107	2025-04-03 13:15:33.14107	(NULL)	$2a$10$tBwKQ/sNnyYkV/xr5L	lifjwolenfd@gmail.com	AAruz	25000	f
40	2025-04-03 20:50:38.09750	2025-04-03 20:50:38.09750	(NULL)	$2a$10$.xgFZMCAm7hHazrU	pranshutest@gmail.com	PranshuTest	25000	f
44	2025-04-05 05:42:09.02343	2025-04-05 05:43:30.32231	(NULL)	$2a$10$F0SxrCODZWwjCckT	s@a.com	Sangam	22590.59960937	f
36	2025-04-03 12:52:12.09679	2025-04-04 14:57:57.22571	(NULL)	$2a$10$wB0EYLaZobHWHzs	guptasang392@gmail.com	Sangam Gupta	9487.94921875	f
41	2025-04-03 21:49:39.68169	2025-04-04 15:23:34.53087	(NULL)	$2a$10$GKQTe.8HOlReRrTfC	sa@i.com	SAmeer Yadav	6827	f
42	2025-04-04 15:36:02.43974	2025-04-04 16:38:59.47848	(NULL)	$2a$10$tCFRvdWznUQ9ZmF	ss@iitk.ac.in	Sameer Yadav	22590.59960937	f
43	2025-04-04 17:35:08.75920	2025-04-04 17:36:34.75642	(NULL)	$2a$10$jZedO9TbMMAvQuni	addy@gmail.com	e	23795.30078125	f
45	2025-04-05 06:43:04.52338	2025-04-05 10:34:15.72635	(NULL)	$2a$10$NJxujfQO55EA7jgElv	aadi1511singh@gmail.com	Aadi Singh	23182.69921875	f
64	2025-04-05 12:04:21.11919	2025-04-05 12:04:21.11919	(NULL)	$2a$10$PqCAEoBMNZJekj3it	xihatac819@bariswc.com	tas	25000	f
47	2025-04-05 08:52:00.26987	2025-04-05 08:52:00.26987	(NULL)	$2a$10$lGm.Cw34HVuRgeMH	ss@hi.com	Sameer	25000	f
51	2025-04-05 09:05:04.56982	2025-04-05 09:05:04.56982	(NULL)	$2a$10$MgB0lRPtNgMg67TL	notdilbarsl234@gmail.com	Dilbar	25000	f
38	2025-04-03 13:15:59.33862	2025-04-05 12:09:28.35596	(NULL)	$2a$10$TR.qAm3Nz2tnl2JStI	aruzawasthi47000@gmail.com	Aruz	21640	f

Unit Name: User Database Testing

This unit test verifies that the users database is functioning as expected. The test ensures that transactions submitted via the Postman client and the frontend queries are correctly updating and reflecting in the database.

Unit Details:

Module: Users Database

The Users database has the following columns:

- id - index of the database
- created_at - the timestamp at which the database entry was created (the timestamp when the user signed up for the first time)
- updated_at - the timestamp at which the database entry was updated (the last login timestamp of the user)
- deleted_at - the timestamp at which the database entry was deleted (when the user permanently deleted his account)
- password - password of the user
- email - email id of the user
- name -name of the user
- balance - balance of the user

Functions Tested: Data submission via Postman and frontend queries, transaction logging, and data retrieval

Test Owner: Dilbar Singh Lamba

Test Date: 04/04/2025

Test Results:

- Transactions initiated through the Postman client were successfully processed and became visible in the database.
- Queries submitted from the frontend accurately retrieved the expected data.
- The database consistently reflected all transaction updates, ensuring that the query results matched the previously made transactions.

Structural Coverage:

- The testing focused on verifying the endpoints used for data submission and retrieval through the Postman client and frontend interface.
- Each query route was exercised to ensure that the correct transaction data is logged and displayed.
- Structural coverage includes validation of data flow from the client to the database and back to the user interface.

3. System Testing

1. **Requirement:** The user should be able to sign in through his email ID and password if he is already a member.

This sign-in functionality underwent thorough testing to ensure its reliability and effectiveness. We created test cases covering various scenarios, including valid and invalid inputs. These test cases were executed by manually navigating to the sign-in page, inputting different combinations of email IDs and passwords, and observing the system's responses. For instance, we tested the successful sign-in process by entering valid credentials and verifying that the user was redirected to the Dashboard page. Additionally, we intentionally input invalid email formats, non-registered user details, and incorrect passwords to validate, then appropriate error messages were displayed in each scenario. We tried signing in without filling both the fields and were prompted to do so. Throughout the testing process, no bugs or issues were encountered, indicating that the sign-in functionality operates as intended and meets the specified requirements.
Test Owner: Pranshu Thirani
Test Date: 31/03/2025
Test Results: The Sign-in page worked as expected. We had tested it on our deployed website. No bug was reported.
Additional Comments: None

2. **Requirement: :** A new user should be able to sign up by creating his account through filling required details (Name, Email, Password).

The sign-up functionality enables new users to create accounts by providing required details, including their name, phone number and email ID along with setting a valid password. To ensure the effectiveness of this functionality, we devised a series of test cases covering various scenarios. These tests were executed by manually navigating to the sign-up page, inputting valid and invalid data into the respective fields, and observing the system's responses. We verified that users could successfully sign up by providing all required details and setting a valid password, leading to the creation of their account. Additionally, we tested scenarios such as entering incomplete or incorrect information, duplicate email IDs, and invalid passwords(less than required min. length) to ensure proper validation and error handling. Throughout the testing process, no other major issues or bugs were encountered, confirming that the sign-up functionality functions as intended and meets the specified requirements.

Test Owner: Aruz Awasthi
Test Date: 31/03/2025
Test Results: The sign-up page worked as expected. .
Additional Comments: None

3. **Requirement:** The Newly registered users get free 25,000 Virtual Balance, which can be viewed under the profile section.

When a new user is registered, 25,000 worth of virtual balance is added to the user's account so that he can start his trading journey with PHInance. This was tested by creating a new user and the balance was verified.

Test Owner: Pranshu Thirani
Test Date: 31/03/2025
Test Results: As expected, 25,000 Virtual Balance was added to the account
Additional Comments: None

4. **Requirement:** The Dashboard contains the list of stocks available, and the watchlist of the user. It also has an "Add to Watchlist" button to add a stock from the list of available stocks to the watchlist of the user.

This test includes the frontend component of the dashboard. The functionality needs to correctly update the watchlist when a stock is added from the list of available stocks. The UI should reflect the updated watchlist.
We acted as a general user and added multiple stocks from the available stocks list to the watchlist. In each case, the watchlist was updated correctly.

Test Owner: Aruz Awasthi
Test Date: 01/04/2025
Test Results: We tested the watchlist functionality on the deployed website, and it worked perfectly fine in all cases.
Additional Comments: None

5. **Requirement:** The Profile Page shall display the user's details, balance, stocks owned, and transaction history.

This test includes both the frontend and backend to fetch and display user data. When a user visits the profile page, their personal details, balance, owned stocks, and transaction

history should be accurately retrieved from the backend and displayed on the frontend. Any changes in the user's portfolio or balance should be reflected in real-time.
We acted as a general user and accessed the profile page. We verified the retrieval of user details, balance, owned stocks, and transaction history. Additionally, we made transactions to check if the updated information is correctly stored in the database and displayed accurately on the profile page.

Test Owner: Aadi Singh
Test Date: 01/04/2025
Test Results: The profile page displayed accurate and updated information in all tested scenarios. Backend APIs were successfully integrated, and the data was fetched and displayed without noticeable delay.
Additional Comments: None

6. **Requirement:** The Profile Page shall include options like basic details, change user details, change password, logout, and permanently delete account.

This test includes both the frontend and backend to ensure proper functionality of various user account options. The frontend should provide a user-friendly interface to access these options, while the backend should handle corresponding requests accurately.
We acted as a general user and tested each option:

Basic Details: Displayed accurately from the database.

Change User Details: Successfully updated details like name, email, and phone number in the backend, and changes were reflected immediately on the profile page.

Change Password: Password was updated correctly after providing the old password and confirming the new one. Password hashing and validation were verified.

Logout: User session was successfully terminated, redirecting to the login page.

Permanently Delete Account: The account was deleted from the database, and the user was logged out and redirected to the homepage. Attempting to log in again with the same credentials failed as expected.

Test Owner: Aadya Dhir
Test Date: 02/04/2025
Test Results: All functionalities worked as intended. Frontend requests were correctly handled by the backend, and responses were consistent and prompt.

Additional Comments: None

7. **Requirement** : There shall be an option to switch between light mode and dark mode on the navigation bar of the website. The navigation bar shall also display the user's name, which, when clicked, opens a dropdown menu containing a "Log out" option that logs the user out.

This test includes both the frontend and backend to ensure proper functionality of the theme toggle and user dropdown.

- **Light/Dark Mode Toggle:** Successfully switched between light and dark themes. The selected theme persisted across page reloads by saving the user's preference in the database.
- **User Name Dropdown:** Clicking on the user's name displayed a dropdown menu with "Log out" option.
- **Log Out Functionality:** Clicking "Log out" properly terminated the user session, clearing tokens and cookies as expected, and redirected to the login page.
- **Persistence Check:** Theme preference remained consistent across different pages and after logging out and logging back in.

Test Owner: Dilbar Singh Lamba
Test Date: 31/03/2025
Test Results: All functionalities worked as intended. Switching themes did not affect other components of the website, and logout functionality was properly implemented.
Additional Comments: Bugs were found related to the dark mode. Some things remained in light mode even after switching to dark mode. These bugs in the frontend were resolved.

8. **Requirement:** The profile page shall display text options "Dashboard" and "Profile". When the user clicks on "Dashboard", they shall be redirected to the dashboard page instantly.

This test includes both frontend and backend components to ensure proper redirection and user interface functionality.

- **Navigation Links Display:** The profile page displayed the "Dashboard" and "Profile" options as intended.
- **Redirection Functionality:** Clicking on "Dashboard" instantly redirected the user to the dashboard page.

- **Backend Handling:** Verified that proper routing was set up to handle the redirection from the profile page to the dashboard.
- **UI Responsiveness:** The navigation options were responsive and displayed correctly across different screen sizes.

Test Owner: Anish Sahu
Test Date: 31/03/2024
Test Results: The redirection from "Profile" to "Dashboard" worked as intended. No delays or errors were encountered during the transition.
Additional Comments: None

9. **Requirement:**The Sell functionality shall work correctly in the “Buy and Sell Stocks” page. Clicking the “Buy” and "Sell" buttons shall open a detailed buy/sell order interface, allowing users to input the number of shares, displays the market value of the stock at which the buy/sell occurs and also shows detailed analysis of the stock using graphs and data values. The changes made by these should be reflected in the profile section.

This test includes both frontend and backend components.

Sell Order Interface: On clicking the "Sell" button, a dedicated sell order panel appeared with all necessary options.
Function Execution: Entering the required details and clicking "Place Sell Order" correctly processed the order, reducing the stock quantity in the user’s portfolio and updating the balance. Similarly “Place Buy Order” increased the stock quantity in the user’s portfolio and updated the balance.
Backend Updates: The database successfully reflected changes in stock ownership, balance adjustments, and transaction history.
Edge Case Handling: Error messages appeared when selling more shares than owned. Selling without sufficient margin displayed appropriate warnings.

Test Owner: Anisha Srivastava
Test Date: 03/04/2025
Test Results: The buy and sell functionalities were tested across multiple scenarios, and all expected behaviors worked correctly.
Additional Comments: Future improvements may include real-time order book updates and estimated profit/loss display before order confirmation.

10. Requirement: The Buy and Sell pages shall provide an option to refresh the prices of stocks, which updates the displayed stock prices to the latest values.

This test includes both frontend and backend components to ensure real-time price updates.

- **Price Refresh Button Display**: The Buy and Sell pages displayed a refresh button labeled "Refresh Prices" as intended.
- **Price Update Functionality:** Clicking the "Refresh Prices" button successfully fetched the latest stock prices from the backend and updated the displayed values.
- **Backend Handling:** Verified that the backend API correctly provided the latest stock prices when requested.
- **UI Responsiveness:** The refresh operation was smooth, and updated prices were displayed within a reasonable response time.

Test Owner: Sangam Gupta
Test Date: 02/04/2025
Test Results: The refresh functionality worked as intended. Prices were updated accurately on each refresh, and no bugs were found.
Additional Comments: Future improvement could include automatic price refresh at regular intervals for real-time monitoring.

11. Requirement: The “Support” page must display Q&A related to trading and PHInance.

The Support page successfully displays Q&A and contacts to get support from the developers.
Test Owner: Anirudh
Test Date: 03/04/2025
Test Results: The Support page works perfectly fine.
Additional Comments: None

12. Requirement: In case the user faces any issue, he should be able to contact the developing team using the **Support Page** and if the user wants any information about the software/development team, he should be able to use the **FAQs** feature to clear them. The "**Contact Us**" feature facilitates seamless communication between users and the development team, allowing users to address any issues, queries, or requests for information about the software or development team. The contact information section prominently displays the team's phone number for easy access.Overall, the "Contact Us" feature enhances user experience by providing a reliable and efficient means of reaching out to the development team for assistance or information.

Test Owner: Aadi Singh
Test Date: 31/03/2025
Test Results: The Contact Us page worked as expected.
Additional Comments: None

13. Requirement: The Total Capital Invested & Total Profit graph on the **Behavioural Analytics** page shall display a user's investment trends and profit changes over time. The graph should provide a clear visual representation of the **Total Capital Invested** and **Total Profit**, allowing users to analyze their financial behavior.

- The graph shall have three time-frame options: Day, Week, and Month.
- The exact numerical values of Total Capital Invested and Total Profit shall be displayed when the user hovers over a data point.
- Data should be fetched in real-time from the backend to reflect accurate investment and profit changes.

This test includes both the frontend and backend functions to fetch and display investment and profit data. The backend is responsible for retrieving real-time investment and profit values, while the frontend ensures smooth visualization through an interactive graph.

We acted as a general user and accessed the Behavioural Analytics page to verify the graph's accuracy. We switched between Day, Week, and Month views to confirm correct data retrieval. Additionally, we tested the hover functionality to ensure exact values were displayed at each data point.

We also simulated transactions that altered the Total Capital Invested and Total Profit, verifying that the updates were accurately reflected on the graph without noticeable delay.

Test Owner: Aadya Dhir
Test Date: 03/04/2025
Test Results:The graph functioned correctly in all tested scenarios.Hovering over data points displayed accurate values.The transition between Day, Week, and Month views was smooth and responsive.Backend APIs correctly retrieved and updated real-time investment and profit data.
Additional Comments: Implementing a comparison feature (e.g., "This Week vs. Last Week") could enhance user insights.

14. Requirement: The Learning Centre must showcase videos on Stock Market Basics, Technical Analysis and AI in Finance.

The test includes selecting and playing every single video and every single quick from each aforementioned category. All the video links were working fine and led to YouTube.com videos on the same topic. We manually checked every link and all of them were functioning.

Test Owner: Anish Sahu
Test Date: 03/04/2025
Test Results: The Videos in the learning center work correctly
Additional Comments: None

15. Requirement: The Learning Centre must have working quizzes on Stock Market Basics, Technical Analysis and AI in Finance.

In the aforementioned categories, Beginner, Intermediate and Advanced quizzes are displayed and are functioning. We opened every single quiz and they were tested on the correct timer (15 minutes), correct number of questions displayed, ability to select answers and display of results. After 15 minutes, the quizzes autocompleted, displaying results including Correct Answers, Total Questions and Score Percentage, alongside the number of coins earnt. Every single quiz was tested and checked for the same. The user is also able to retake the test to check their improvement (the user only earns the increment in coins, which is capped at a certain number for each quiz).

Test Owner: Anisha Srivastava
Test Date: 02/04/2025
Test Results: All the quizzes were functioning fine and the user was able to retake them.
Additional Comments: More quizzes on specialized topics in Finance and trading can be added to improve the learning experience.

16. Requirement: The Coin system in the learning center must work fine, granting the user ‘coins’ for watching videos and/or solving quizzes.

The coins of the user were being incremented after watching a video by the amount specified on the video. This was tested for all video on the website. Further, coins were also added proportionally to the score the user achieved in certain quizzes. The coin system was tested thoroughly and was checked to be working fine.

Test Owner: Sangam Gupta
Test Date: 03/04/2025
Test Results: The coin system in the learning center is working fine.

Additional Comments: Currently, the coins serve as a way to gamify the process of learning. Additional improvements can be made in terms of the tangible use and benefit of these coins.

17. Requirement: The Behavioral Analysis page must correctly depict analysis of the user's portfolio and their trajectory.

The Behaviour Analysis Page correctly shows multiple graphs:

1: A Graph showing the total money invested and the total profit earned on a Day/Week/Month scale. The Graph updates every day with latest information about the user's portfolio.
2: The Page shows a Pie chart demarcating Winning Trades, Losing Trades, High-Risk Trades and Low-Risk trades done by the user, which also updates dynamically.
3: The Page shows a Profit & Loss Trendline, showing the update their net overall profit and loss on a time scale. This too updates dynamically.
4: The Page correctly displays a pie chart displaying the categories of investment of the user, divided into Stocks, Crypto, Forex and Options. This too updates dynamically.

Test Owner: Sameer Yadav
Test Date: 03/04/2025
Test Results: The Behavioural Analysis illustrations and statistics of the user are working fine.
Additional Comments: Further improvements in the metrics of behavioural analysis could be made to improve user experience and learning.

18. Requirement: The Portfolio Management page must list down the constituent stocks of the user's portfolio alongside their current standings. It must also show when stock purchases were made, their contemporary rate and price.

The Portfolio Management page correctly displays all the stocks that are part of the user's portfolio, their current market value, the returns made and their valuation. In a drop down menu, it also showcases dates of purchase of the stock and their price at the time. The page updates with changes to the user's portfolio in terms of buying and selling stocks. It also displays the average price at which the stock was bought over a long period of time.

Test Owner: Dilbar Singh Lamba
Test Date: 03/04/2025

Test Results: The Portfolio Management page seems to be working fine and displaying the contents of the portfolio correctly.
Additional Comments: None

19. Requirement: The Risk Management page must display a detailed Risk Analysis of the user's portfolio.

The Risk Management page displays a Risk analysis of the user's portfolio. We tested it by adding new stocks to the portfolio. The Page serves information about the stock, like its Average Price, Quantity of stock owned, 1-Day VAR and 1 Month VAR. These attributes of the stocks are regularly updated and the page worked well on all the tests made with a plethora of stocks.

Test Owner: Sameer Yadav
Test Date: 03/04/2025
Test Results: The Risk Management page is working successfully.
Additional Comments: None

4. Conclusion

How Effective and exhaustive was the testing?

Testing was done almost exhaustively. Each API underwent testing to ensure that they were modifying the database and returning the desired information when needed. We have taken care of all cases, thus ensuring complete branch coverage. The frontend was also tested and bugs were resolved. Validity of each input has been checked in all forms. Appropriate alerts have been shown in such cases. Edge cases were also tested and were confirmed to work as expected.The testing was quite effective. A developer tested a component that was not developed by him/her.

Which components have not been tested adequately?

The AI-powered predictions page has not been tested adequately. This is because the AI model required for generating predictions was taking over 12 hours to run and fetch the necessary data, which made it infeasible to test within the available time frame.

What difficulties have you faced during testing?

The main difficulty faced during testing was ensuring non-functional requirements are satisfied, in system testing.

How could the testing process be improved?

The testing could have been improved by doing the unit testing or integration testing as automated rather than manual. The developer might subconsciously assume a few crucial things which may be not so obvious to an end-user or someone who was not involved in the development of the software

Appendix A - Group Log

Date	Timing	Duration	Agenda
31/03/2025	14:00 - 16:30	2.5 hrs	● Discussed various types of testing. ● Distributed the tasks of various types of testing and the user manual among subgroups within the team.
01/04/2025	18:00 - 19:30	1.5 hrs	● Discussed the various doubts faced by each subgroup in their respective tasks.
02/04/2025	17:00 - 19:00	2 hrs	● Every subgroup updated each other with their work. ● Discussed doubts among team members.
04/04/2025	16:00 - 20:00 21:00 - 23:00	6 hrs	● Finished the remaining work and reviewed the testing document. after clearing various doubts in the testing document. ● Finalised the testing document.