DATASPHERE PROJECT PRESENTATION

PRESENTED BY: COHORT B GROUP 18

DELICE ISHIMWE
SUSANNA AGYAPONG
SINAM AMETEWEE
LYDIA AMOAKOAA

Our Team and Roles

Project Manager

Lydia Amoakoaa

Front-end Developer

Sinam Ametewee

Back-end Developer

Susanna Agyapong

Back-end Developer

Delice Ishimwe

Background

PROBLEM

Companies struggle with streamlining feedback through emails, call among others. Thus, they struggle to track insights, detect trends, and improve services effectively.

There is a need for a centralized and intelligent system that can collect, organize, and analyze customer feedback in real-time to support data-driven decision-making and continuous improvement.

The Solution - DataSphere

We are creating a web-based platform for collecting, managing, and analyzing customer feedback.

Provides a centralized system to eliminate inefficiencies and fragmented approaches

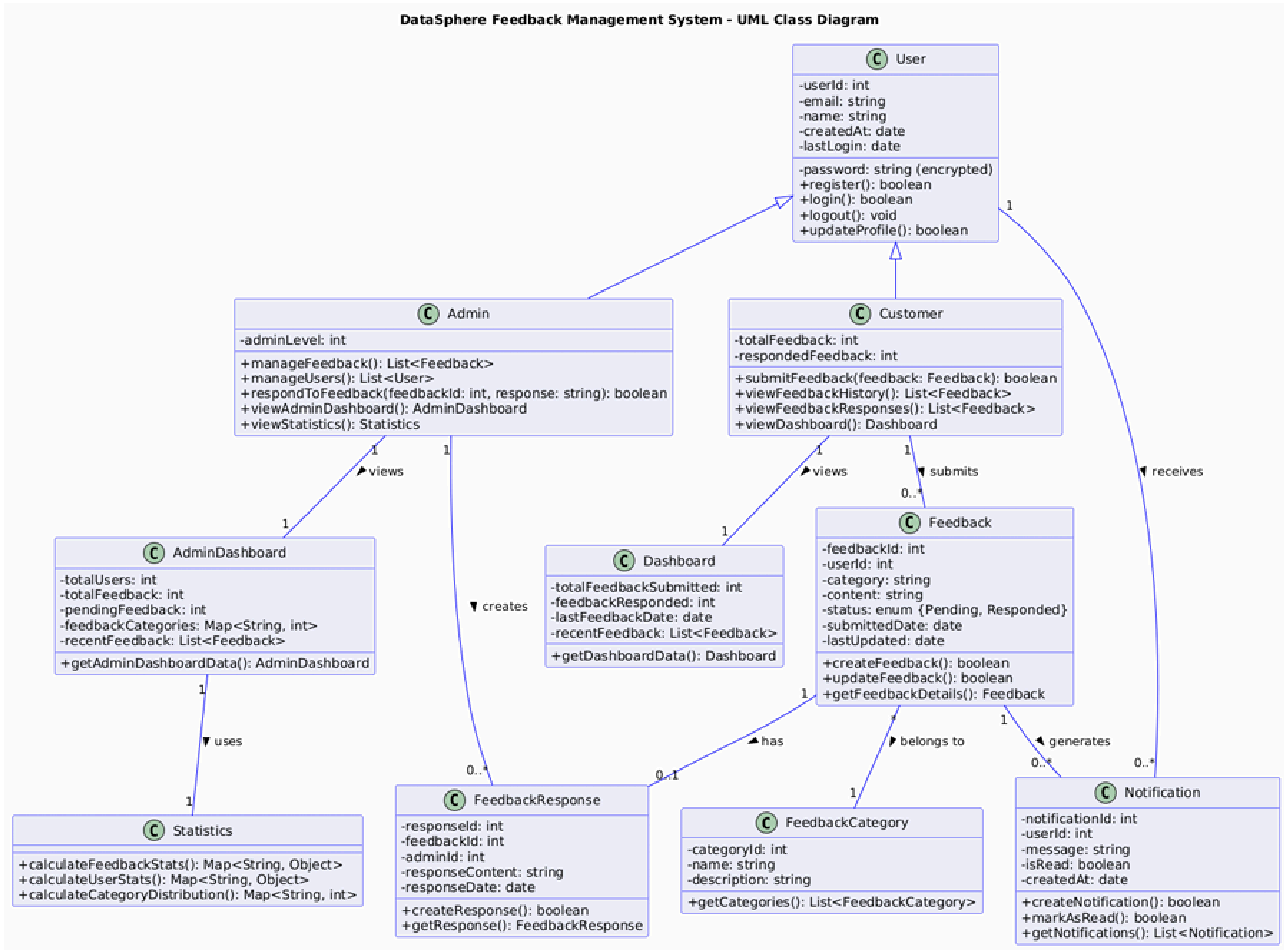
Enables real-time analytics for better decision-making and improved service delivery.

Key Features of DataSphere

- Centralized Feedback Collection: Streamlines feedback submission and management.
- Real-Time Analytics: Visualizes trends, ratings, and engagement metrics.
- Role-Based Access: Secure logins and access control for customers and admins.
- Interactive Dashboards: Provides feedback history for customers and key metrics for admins.
- Feedback Responses: Admins can directly respond to feedback and manage interactions efficiently.
- Pending Feedback Updates: The system continuously updates and processes feedback that has not yet been responded to, ensuring real-time tracking and management of unresolved feedback.
- Scalable and Secure: Supports high user loads and ensures data security.

UML Class diagram

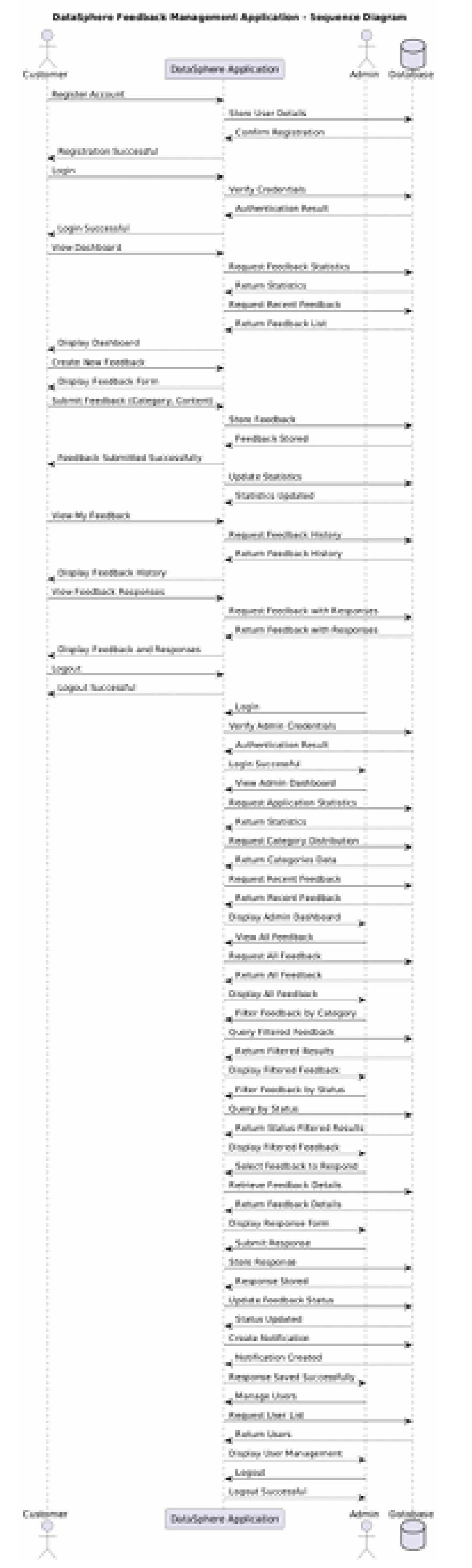
[Click here to vew the UML Class Diagram]


DataSphere Feedback Management System - UML Class Diagram
User
-userId: int
-email: string
-name: string
-createdAt: date
-lastLogin: date
-password: string (encrypted)
+register(): boolean
+login(): boolean
+logout(): void
+updateProfile(): boolean
Admin
-adminLevel: int
+manageFeedback(): List<Feedback>
+manageUsers(): List<User>
+respondToFeedback(feedbackId: int, response: string): boolean
+viewAdminDashboard(): AdminDashboard
+viewStatistics(): Statistics
Customer
-totalFeedback: int
-respondedFeedback: int
+submitFeedback(feedback: Feedback): boolean
+viewFeedbackHistory(): List<Feedback>
+viewFeedbackResponses(): List<Feedback>
+viewDashboard(): Dashboard
AdminDashboard
-totalUsers: int
-totalFeedback: int
-pendingFeedback: int
-feedbackCategories: Map<String, int>
-recentFeedback: List<Feedback>
+getAdminDashboardData(): AdminDashboard
Dashboard
-totalFeedbackSubmitted: int
-feedbackResponded: int
-lastFeedbackDate: date
-recentFeedback: List<Feedback>
+getDashboardData(): Dashboard
Feedback
-feedbackId: int
-userId: int
-category: string
-content: string
-status: enum {Pending, Responded}
-submittedDate: date
-lastUpdated: date
+createFeedback(): boolean
+updateFeedback(): boolean
+getFeedbackDetails(): Feedback
Statistics
+calculateFeedbackStats(): Map<String, Object>
+calculateUserStats(): Map<String, Object>
+calculateCategoryDistribution(): Map<String, int>
FeedbackResponse
-responseId: int
-feedbackId: int
-adminId: int
-responseContent: string
-responseDate: date
+createResponse(): boolean
+getResponse(): FeedbackResponse
FeedbackCategory
-categoryId: int
-name: string
-description: string
+getCategories(): List<FeedbackCategory>
Notification
-notificationId: int
-userId: int
-message: string
-isRead: boolean
-createdAt: date
+createNotification(): boolean
+markAsRead(): boolean
+getNotifications(): List<Notification>
1 views 1
1 creates 0..*
1 views 1
1 submits 0..*
1 receives 0..*
1 uses 1
1 has 0..1
* belongs to 1
1 generates 0..*

Sequence diagram

Our Sequence Diagram outlines the flow of interactions between components of the system over time. It demonstrates how a customer submits feedback, how the system processes the input, notifies the admin, and how the admin responds through the platform. This diagram helps visualize the chronological order of events, ensuring smooth communication between system modules and aiding in backend implementation.

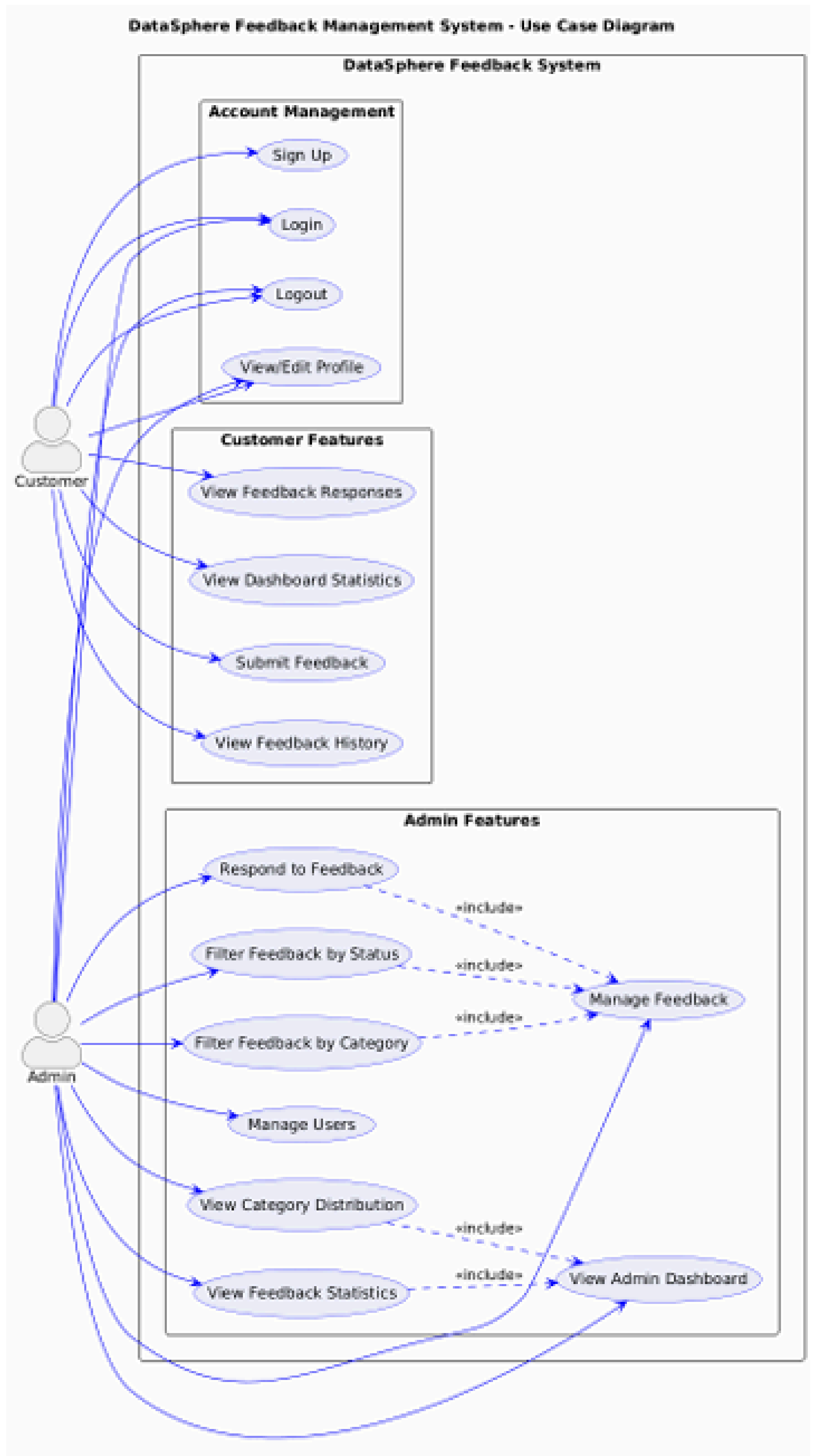
Click here to vew the Sequence Diagram



Use-Case diagram

The Use Case Diagram shows the key actors and their interactions with the DataSphere platform. It highlights the different functionalities available to both customers and administrators, such as submitting feedback, viewing feedback responses, and managing feedback data.

[Click here to view the Use Case Diagram]()



Technologies and frameworks

We used the following technologies and frameworks for our project:

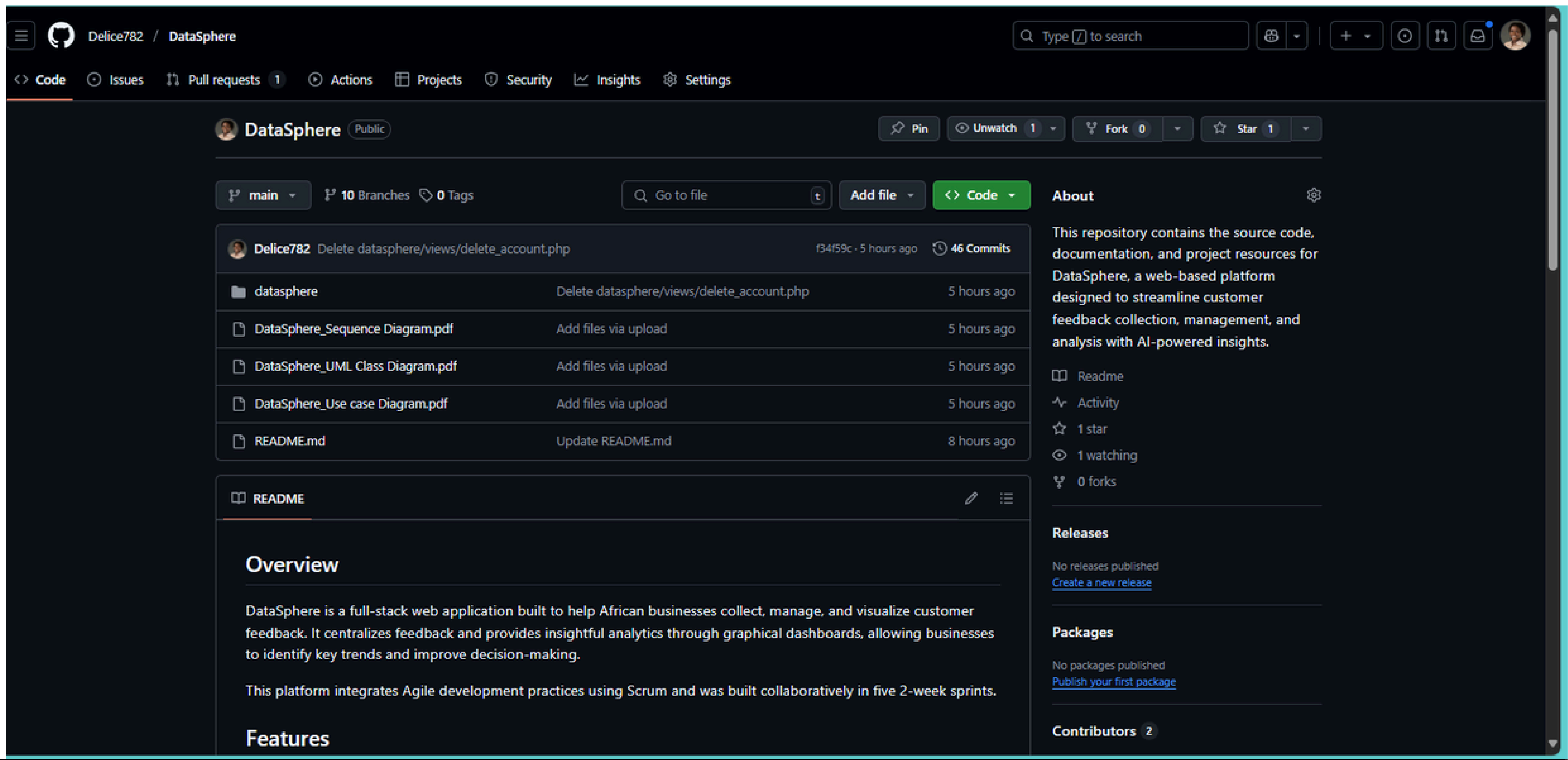
- Frontend: HTML, CSS, Javascript, Bootstrap, Font Awesome
- Backend: PHP
- Database: MySQL
- Project Management: Jira (5 two-weeks Sprints)
- Version Control: Git & GitHub

Github

We used GitHub as our central platform for version control, collaboration, and documentation. It allowed us to manage code changes efficiently through branches and pull requests, track issues and assign tasks, and maintain a clear project history.

Our README file provides detailed setup instructions and usage guidelines, ensuring the project is easy to understand and contribute.

[Click here to view GitHub repository]()



Delice782 Delete datasphere/views/delete_account.php	f34f59c · 5 hours ago	46 Commits
datasphere	Delete datasphere/views/delete_account.php	5 hours ago
DataSphere_Sequence Diagram.pdf	Add files via upload	5 hours ago
DataSphere_UML Class Diagram.pdf	Add files via upload	5 hours ago
DataSphere_Use case Diagram.pdf	Add files via upload	5 hours ago
README.md	Update README.md	8 hours ago

README

Overview

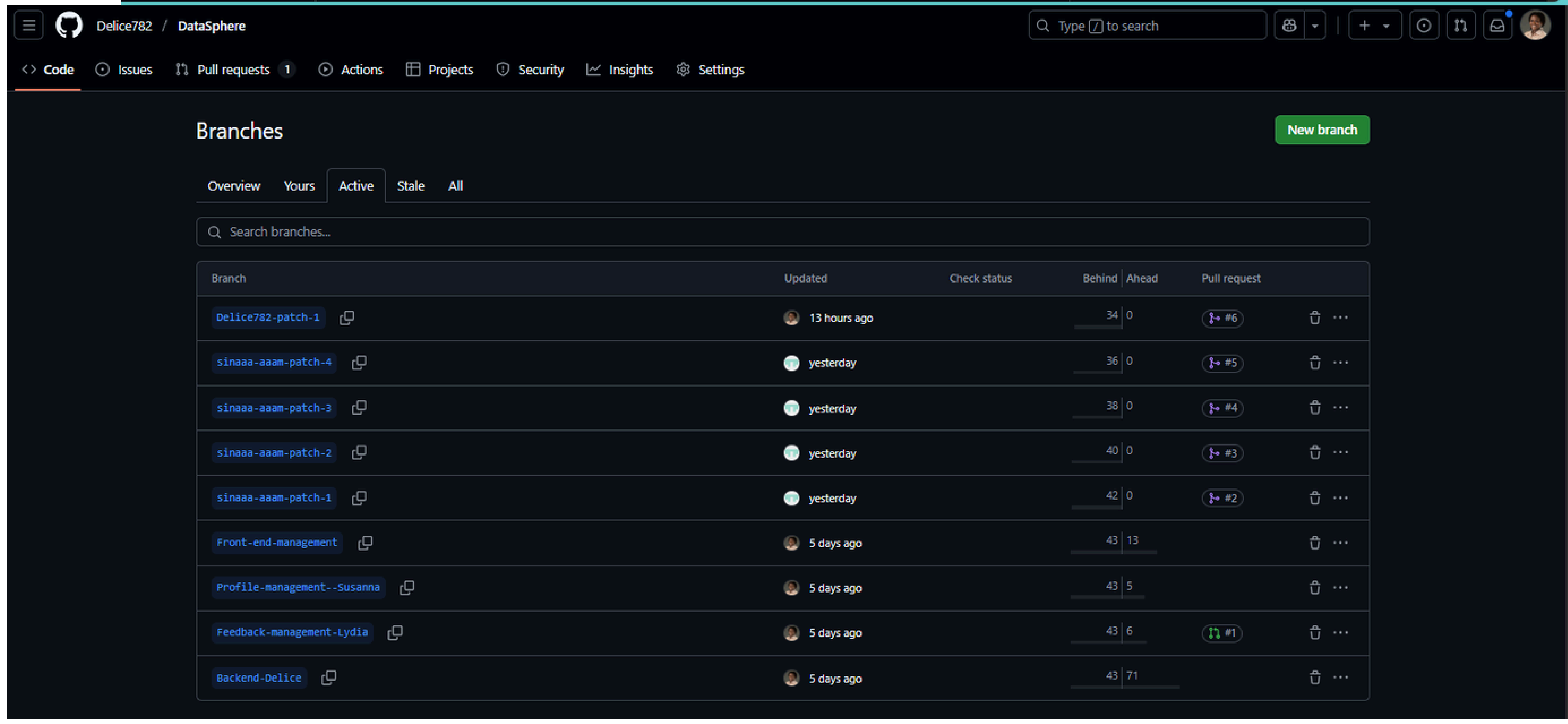
DataSphere is a full-stack web application built to help African businesses collect, manage, and visualize customer feedback. It centralizes feedback and provides insightful analytics through graphical dashboards, allowing businesses to identify key trends and improve decision-making.

This platform integrates Agile development practices using Scrum and was built collaboratively in five 2-week sprints.

Features

About

This repository contains the source code, documentation, and project resources for DataSphere, a web-based platform designed to streamline customer feedback collection, management, and analysis with AI-powered insights.

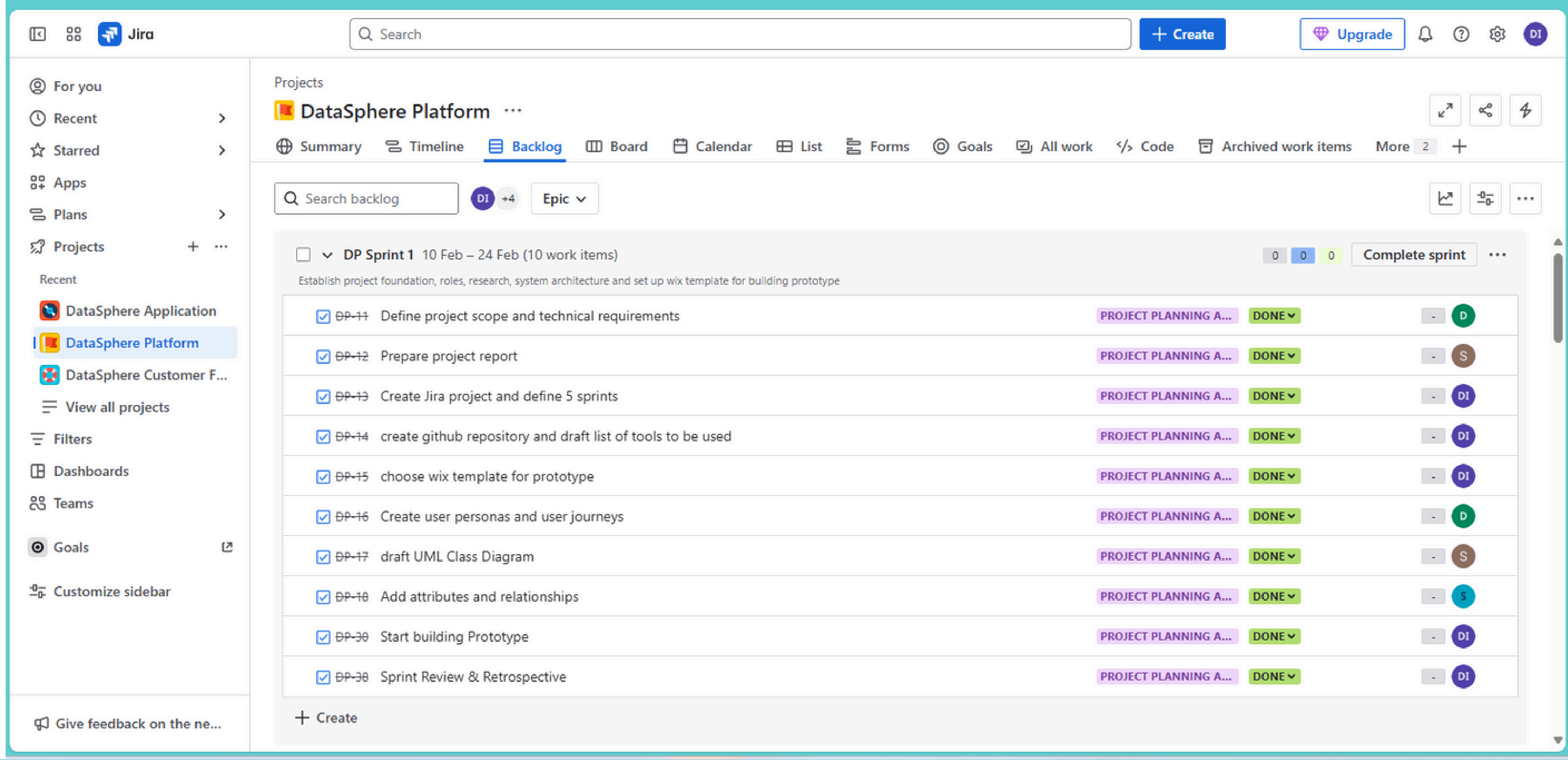


Branch	Updated	Check status	Behind	Ahead	Pull request
Delice782-patch-1	13 hours ago		34	0	#6
sinaaa-aaam-patch-4	yesterday		36	0	#5
sinaaa-aaam-patch-3	yesterday		38	0	#4
sinaaa-aaam-patch-2	yesterday		40	0	#3
sinaaa-aaam-patch-1	yesterday		42	0	#2
Front-end-management	5 days ago		43	13	
Profile-management--Susanna	5 days ago		43	5	
Feedback-management-Lydia	5 days ago		43	6	#1
Backend-Delice	5 days ago		43	71	

JIRA

For our scrum processes, we used the help of Jira to help us set 5 sprints, each lasting 2 weeks and also assign tasks to every team member for smooth collaboration throughout the journey of the project.

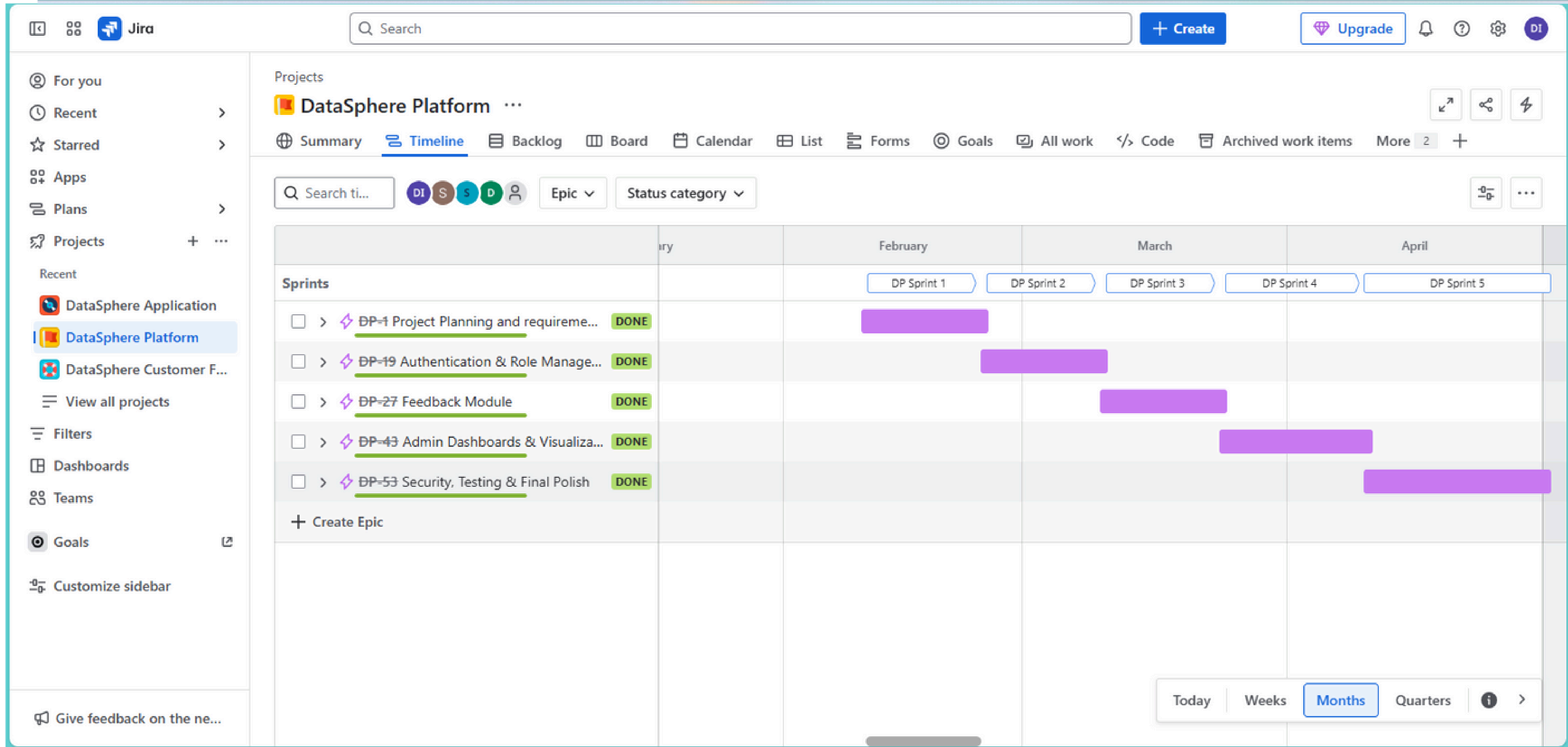
Click here to view our Backlog



DP Sprint 1 10 Feb – 24 Feb (10 work items)

Establish project foundation, roles, research, system architecture and set up wix template for building prototype

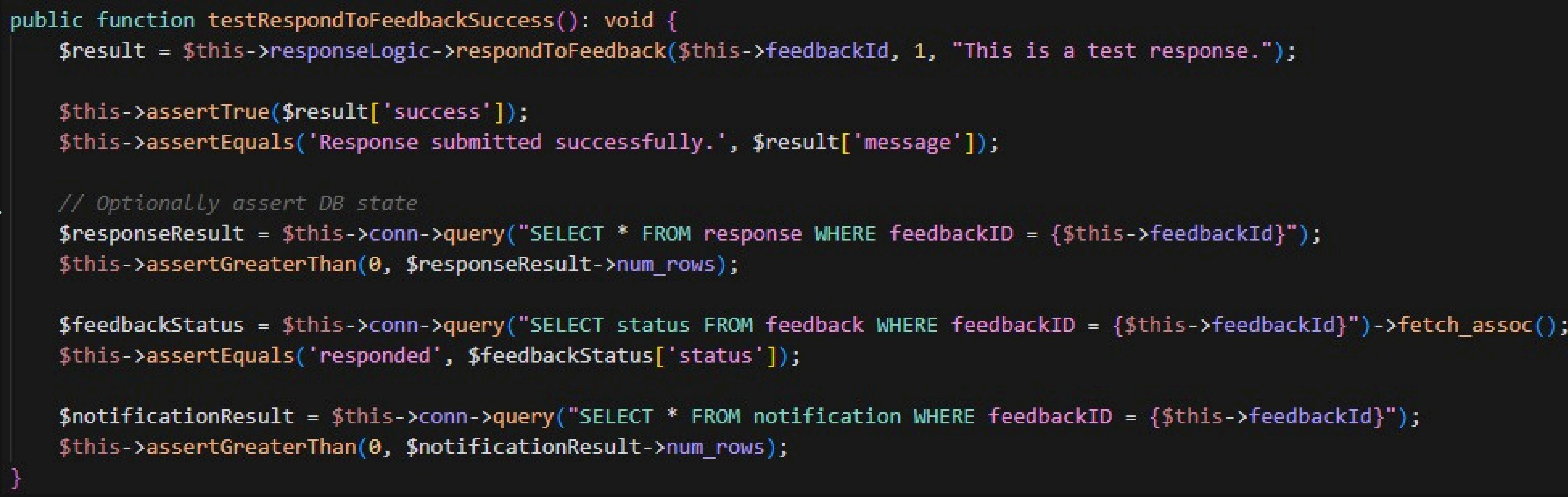
~~DP-11~~	Define project scope and technical requirements	PROJECT PLANNING A...	DONE	D
~~DP-12~~	Prepare project report	PROJECT PLANNING A...	DONE	S
~~DP-13~~	Create Jira project and define 5 sprints	PROJECT PLANNING A...	DONE	DI
~~DP-14~~	create github repository and draft list of tools to be used	PROJECT PLANNING A...	DONE	DI
~~DP-15~~	choose wix template for prototype	PROJECT PLANNING A...	DONE	DI
~~DP-16~~	Create user personas and user journeys	PROJECT PLANNING A...	DONE	D
~~DP-17~~	draft UML Class Diagram	PROJECT PLANNING A...	DONE	S
~~DP-18~~	Add attributes and relationships	PROJECT PLANNING A...	DONE	S
~~DP-30~~	Start building Prototype	PROJECT PLANNING A...	DONE	DI
~~DP-38~~	Sprint Review & Retrospective	PROJECT PLANNING A...	DONE	DI



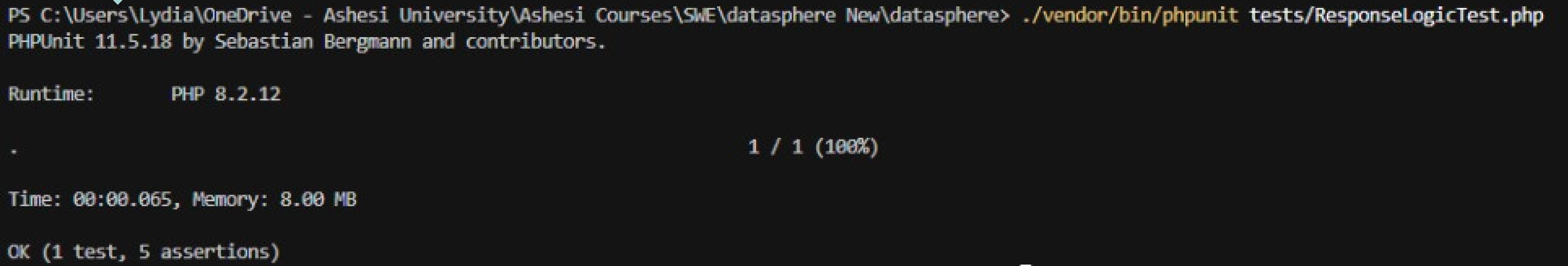
~~DP-1~~	Project Planning and requireme...	DONE
~~DP-19~~	Authentication & Role Manage...	DONE
~~DP-27~~	Feedback Module	DONE
~~DP-43~~	Admin Dashboards & Visualiza...	DONE
~~DP-53~~	Security, Testing & Final Polish	DONE

TESTS

RESULTS



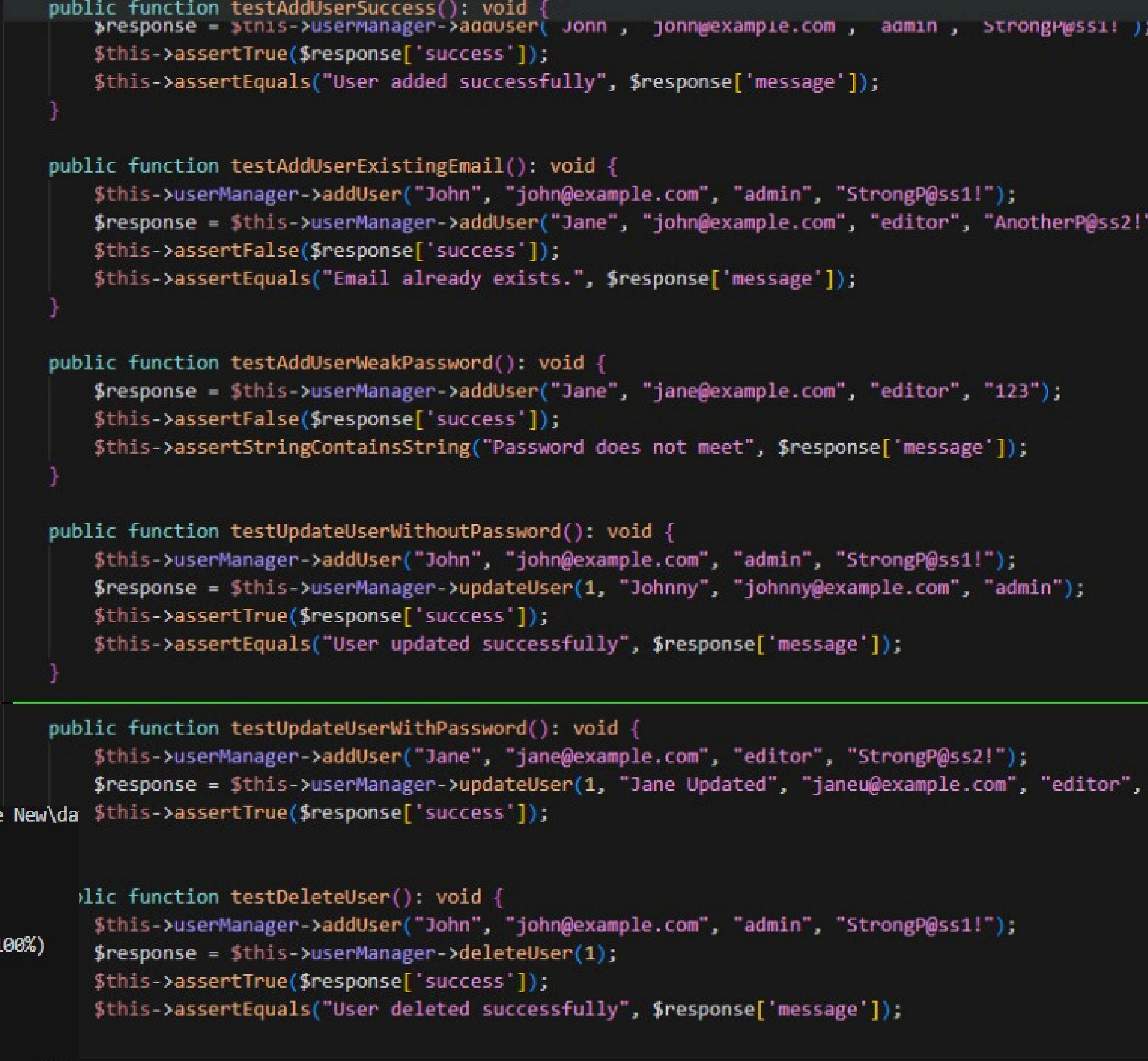
```php
public function testRespondToFeedbackSuccess(): void {
    $result = $this->responseLogic->respondToFeedback($this->feedbackId, 1, "This is a test response.");

    $this->assertTrue($result['success']);
    $this->assertEquals('Response submitted successfully.', $result['message']);

    // Optionally assert DB state
    $responseResult = $this->conn->query("SELECT * FROM response WHERE feedbackID = {$this->feedbackId}");
    $this->assertGreaterThan(0, $responseResult->num_rows);

    $feedbackStatus = $this->conn->query("SELECT status FROM feedback WHERE feedbackID = {$this->feedbackId}")->fetch_assoc();
    $this->assertEquals('responded', $feedbackStatus['status']);

    $notificationResult = $this->conn->query("SELECT * FROM notification WHERE feedbackID = {$this->feedbackId}");
    $this->assertGreaterThan(0, $notificationResult->num_rows);
}
```



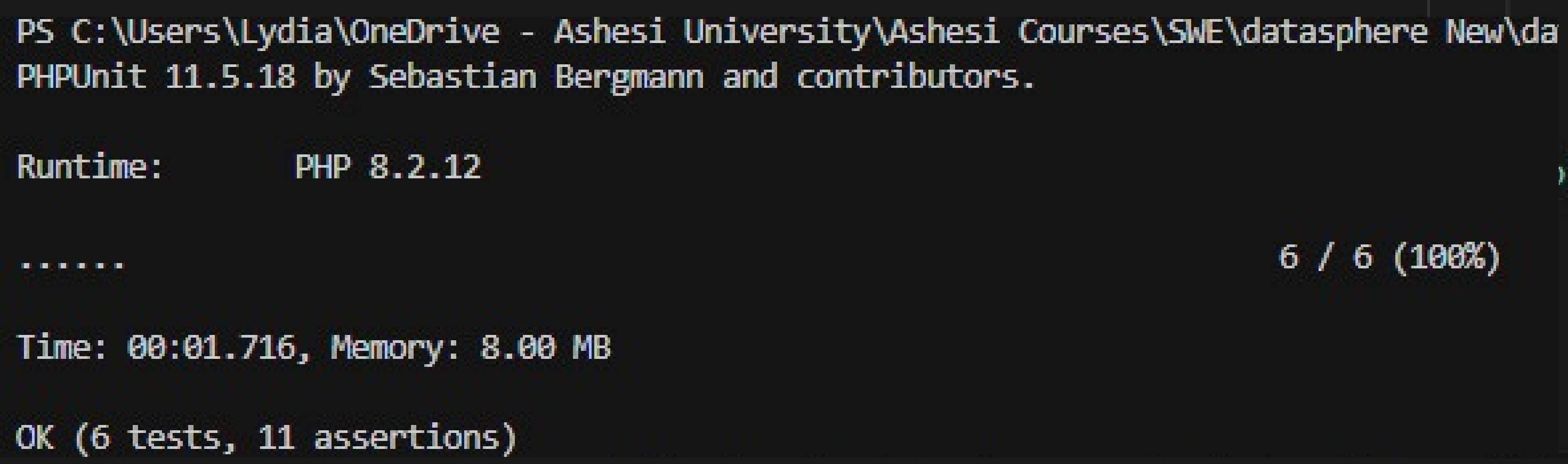
```
PS C:\Users\Lydia\OneDrive - Ashesi University\Ashesi Courses\SWE\datasphere New\datasphere> ./vendor/bin/phpunit tests/ResponseLogicTest.php
PHPUnit 11.5.18 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12

.                                                                   1 / 1 (100%)

Time: 00:00.065, Memory: 8.00 MB

OK (1 test, 5 assertions)
```

TESTS

RESULTS

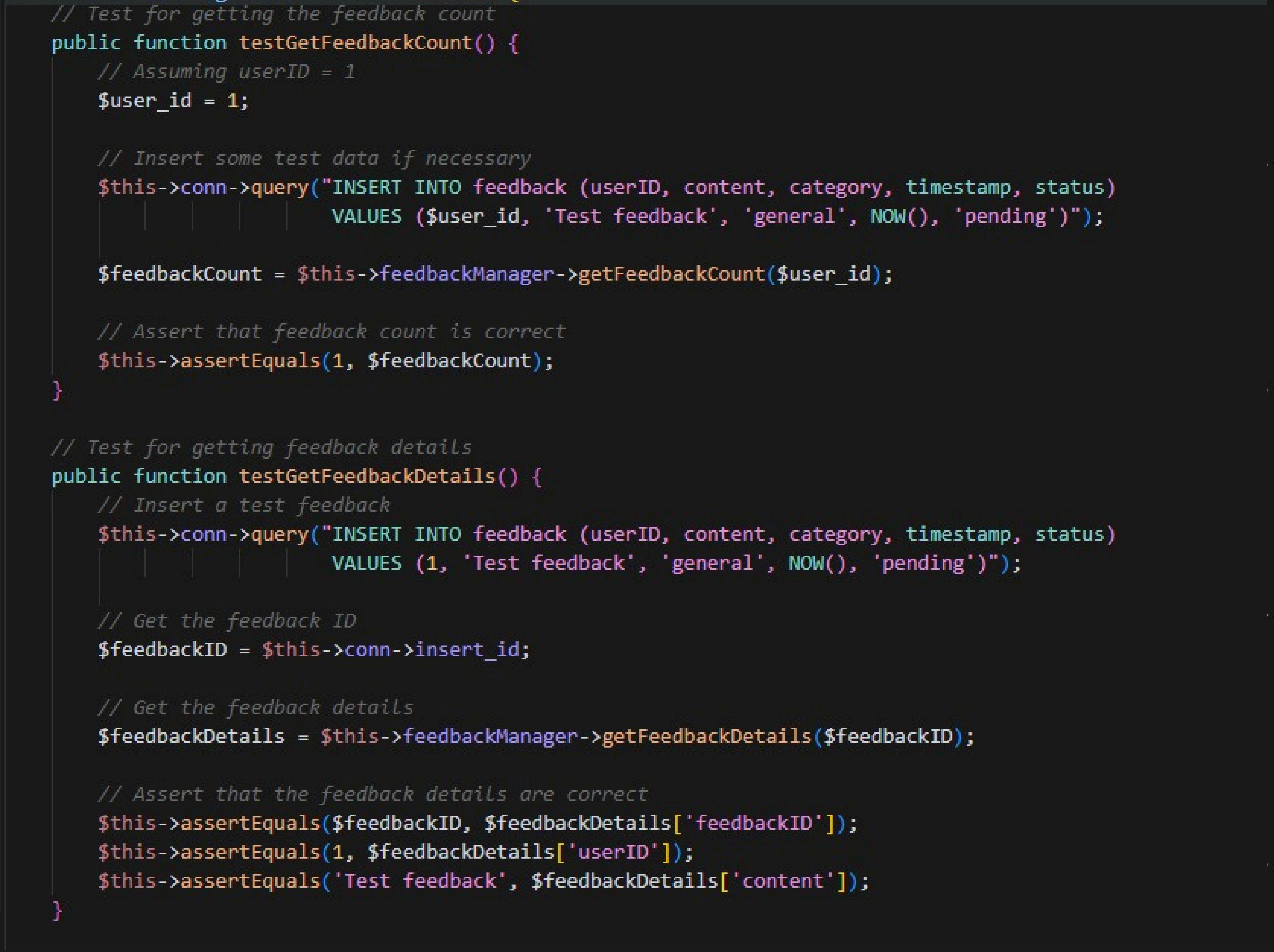


```
public function testAddUserSuccess(): void {
    $response = $this->userManager->addUser("John", "john@example.com", "admin", "StrongP@ss1!");
    $this->assertTrue($response['success']);
    $this->assertEquals("User added successfully", $response['message']);
}

public function testAddUserExistingEmail(): void {
    $this->userManager->addUser("John", "john@example.com", "admin", "StrongP@ss1!");
    $response = $this->userManager->addUser("Jane", "john@example.com", "editor", "AnotherP@ss2!"
    $this->assertFalse($response['success']);
    $this->assertEquals("Email already exists.", $response['message']);
}

public function testAddUserWeakPassword(): void {
    $response = $this->userManager->addUser("Jane", "jane@example.com", "editor", "123");
    $this->assertFalse($response['success']);
    $this->assertStringContainsString("Password does not meet", $response['message']);
}

public function testUpdateUserWithoutPassword(): void {
    $this->userManager->addUser("John", "john@example.com", "admin", "StrongP@ss1!");
    $response = $this->userManager->updateUser(1, "Johnny", "johnny@example.com", "admin");
    $this->assertTrue($response['success']);
    $this->assertEquals("User updated successfully", $response['message']);
}

public function testUpdateUserWithPassword(): void {
    $this->userManager->addUser("Jane", "jane@example.com", "editor", "StrongP@ss2!");
    $response = $this->userManager->updateUser(1, "Jane Updated", "janeu@example.com", "editor",
    $this->assertTrue($response['success']);

lic function testDeleteUser(): void {
    $this->userManager->addUser("John", "john@example.com", "admin", "StrongP@ss1!");
    $response = $this->userManager->deleteUser(1);
    $this->assertTrue($response['success']);
    $this->assertEquals("User deleted successfully", $response['message']);
```

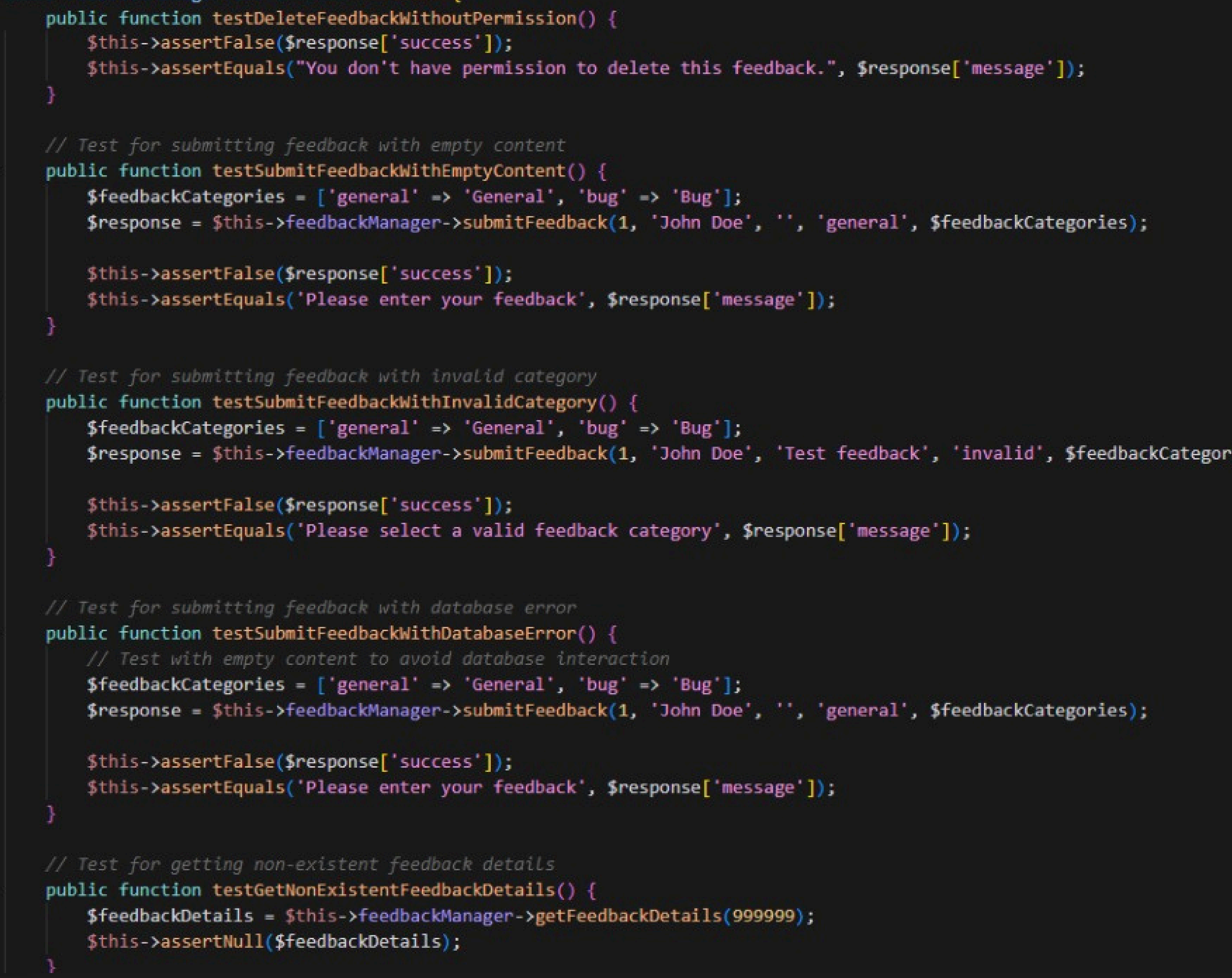


```
PS C:\Users\Lydia\OneDrive - Ashesi University\Ashesi Courses\SWE\datasphere New\da
PHPUnit 11.5.18 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12

......                                                          6 / 6 (100%)

Time: 00:01.716, Memory: 8.00 MB

OK (6 tests, 11 assertions)
```

TESTS

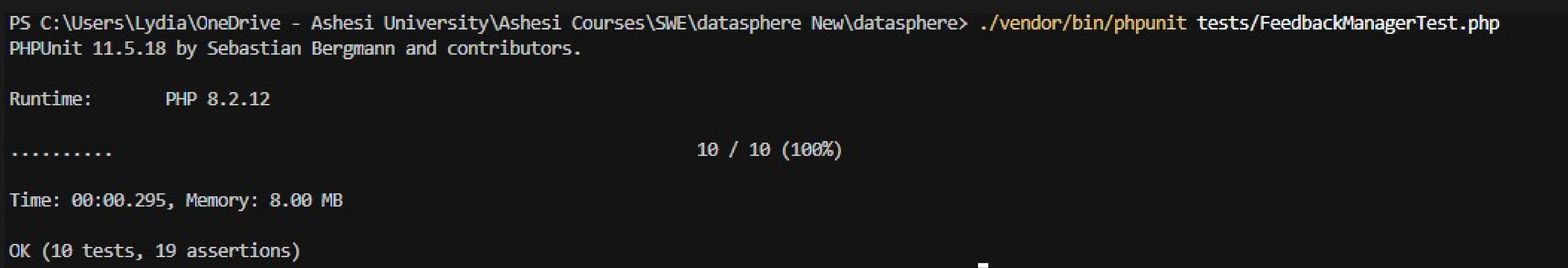


```php
// Test for getting the feedback count
public function testGetFeedbackCount() {
    // Assuming userID = 1
    $user_id = 1;

    // Insert some test data if necessary
    $this->conn->query("INSERT INTO feedback (userID, content, category, timestamp, status)
                        VALUES ($user_id, 'Test feedback', 'general', NOW(), 'pending')");

    $feedbackCount = $this->feedbackManager->getFeedbackCount($user_id);

    // Assert that feedback count is correct
    $this->assertEquals(1, $feedbackCount);
}

// Test for getting feedback details
public function testGetFeedbackDetails() {
    // Insert a test feedback
    $this->conn->query("INSERT INTO feedback (userID, content, category, timestamp, status)
                        VALUES (1, 'Test feedback', 'general', NOW(), 'pending')");

    // Get the feedback ID
    $feedbackID = $this->conn->insert_id;

    // Get the feedback details
    $feedbackDetails = $this->feedbackManager->getFeedbackDetails($feedbackID);

    // Assert that the feedback details are correct
    $this->assertEquals($feedbackID, $feedbackDetails['feedbackID']);
    $this->assertEquals(1, $feedbackDetails['userID']);
    $this->assertEquals('Test feedback', $feedbackDetails['content']);
}
```



```php
public function testDeleteFeedbackWithoutPermission() {
    $this->assertFalse($response['success']);
    $this->assertEquals("You don't have permission to delete this feedback.", $response['message']);
}

// Test for submitting feedback with empty content
public function testSubmitFeedbackWithEmptyContent() {
    $feedbackCategories = ['general' => 'General', 'bug' => 'Bug'];
    $response = $this->feedbackManager->submitFeedback(1, 'John Doe', '', 'general', $feedbackCategories);

    $this->assertFalse($response['success']);
    $this->assertEquals('Please enter your feedback', $response['message']);
}

// Test for submitting feedback with invalid category
public function testSubmitFeedbackWithInvalidCategory() {
    $feedbackCategories = ['general' => 'General', 'bug' => 'Bug'];
    $response = $this->feedbackManager->submitFeedback(1, 'John Doe', 'Test feedback', 'invalid', $feedbackCategor

    $this->assertFalse($response['success']);
    $this->assertEquals('Please select a valid feedback category', $response['message']);
}

// Test for submitting feedback with database error
public function testSubmitFeedbackWithDatabaseError() {
    // Test with empty content to avoid database interaction
    $feedbackCategories = ['general' => 'General', 'bug' => 'Bug'];
    $response = $this->feedbackManager->submitFeedback(1, 'John Doe', '', 'general', $feedbackCategories);

    $this->assertFalse($response['success']);
    $this->assertEquals('Please enter your feedback', $response['message']);
}

// Test for getting non-existent feedback details
public function testGetNonExistentFeedbackDetails() {
    $feedbackDetails = $this->feedbackManager->getFeedbackDetails(999999);
    $this->assertNull($feedbackDetails);
}
```

RESULTS



```
PS C:\Users\Lydia\OneDrive - Ashesi University\Ashesi Courses\SWE\datasphere New\datasphere> ./vendor/bin/phpunit tests/FeedbackManagerTest.php
PHPUnit 11.5.18 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12

..........                                                        10 / 10 (100%)

Time: 00:00.295, Memory: 8.00 MB

OK (10 tests, 19 assertions)
```

Powering Our Solution in the Cloud

Deployed on Microsoft Azure Virtual Machine.

Scales seamlessly as user demands grow.

Ensures robust performance under real-world conditions.

Bringing the DataSphere Feedback Management System to life.

Challenges Faced

- Communication Gaps in Daily Stand-Ups
- We faced issues with merging code and synchronizing team efforts.
- Deploying and configuring the system on Azure’s virtual machine was challenging.

Key Learnings

Proper planning minimizes scope creep and ensures focused progress.

Regular, intentional communication keeps the team on track and minimizes misunderstandings.

Enforcing consistent version control prevents conflicts and streamlines the development process.

Early testing and strategic continuous integration result in fewer issues and smoother deployments.

THE DEMO

DATASPHERE PLATFORM:

http://172.174.224.159/DataSphere/datasphere

To access and test the admin side, use the following credentials:
- Email: utosun@gmail.com
- Password: UmutTosun@123

PROJECT DEMO VIDEO:

https://drive.google.com/file/d/1mp-bFmK6zfAhwguxl4PlhBSTsRrmnrYY/view?usp=sharing

Thank you!

DataSphere

Smart Insights, Smarter Decisions