ASHESI UNIVERSITY

DEVELOPING A WEB APPLICATION: DATASPHERE

SOFTWARE ENGINEERING FINAL PROJECT

Cohort B Group 18

Delice Ishimwe

Sinam Ametewee

Lydia Amoakoaa

Susanna Agyapong

Lecturer: Dr. Umut Tosun

May, 2025

Table of Contents

1. Project Overview

1.1 Project Background

Customer feedback is a vital component of business success, enabling companies to refine their services, address concerns, and maintain a competitive edge. However, African businesses struggle to effectively collect and utilize customer feedback due to fragmented and inefficient processes.

A comprehensive study by Hinson et al. (2020), highlights that modern African consumers are no longer passive recipients but active participants in the product and service experience. Despite this shift, most African businesses still rely on disconnected feedback collection methods such as manual surveys, scattered email communications, and informal feedback channels. Dr. Michélle Booysen’s (2025) analysis of African business processes further emphasizes that these outdated approaches create substantial data gaps, making it difficult for businesses to consolidate insights, detect critical trends, and implement timely improvements. Without a streamlined system, companies waste valuable time piecing together feedback from multiple sources, leading to slow response times, inconsistent data collection, and inefficient resource allocation. These inefficiencies weaken customer satisfaction and limit the ability of African businesses to compete in an increasingly digital and customer-driven marketplace.

1.2 Problem Statement

The fragmented nature of customer feedback management in African businesses leads to delayed response times, inconsistent data collection, and inefficient resource utilization. Companies struggle to consolidate feedback from various sources, making it difficult to identify critical insights, improve service quality, and meet customer expectations. This lack of an integrated feedback system ultimately reduces customer satisfaction and limits businesses' ability to adapt to evolving consumer needs.

1.3 Solution

To address these challenges, we propose DataSphere, a web-based platform which would be designed to streamline the collection, management, and analysis of customer feedback for African businesses. DataSphere provides an intuitive interface where customers can easily submit feedback, and companies can efficiently track, analyze, and visualize customer insights in real time.

By centralizing feedback collection, DataSphere eliminates inefficiencies caused by fragmented approaches such as manual surveys, emails, and informal feedback channels. The platform enhances decision-making by providing structured data analytics, helping businesses identify customer needs, respond faster, and improve service delivery.

With DataSphere, African businesses can transition from reactive to proactive customer engagement, fostering stronger relationships and improving overall competitiveness in the digital marketplace.

1.4 Project Objectives

In order for use to be able to execute our plans accordingly and not have a project creep we decided to outline the following objectives:

- Develop a web-based platform that centralizes customer feedback collection and management for improved decision-making.
- Design and implement customer and admin dashboards for tracking feedback submissions, responses, and engagement metrics.
- Ensure secure user authentication and data handling to protect sensitive customer information.
- Enable administrators to respond directly to feedback, manage customer interactions efficiently.

- Integrate a notification system to keep users informed about feedback responses and updates.
- Implement visualization tools to help companies track trends and make data-driven decisions.
- Optimize system performance and security to ensure scalability and reliability for long-term usage.

1.5 Key Features of DataSphere

The DataSphere system includes the following key features:

- User Authentication & Authorization: A secure login system with role-based access to ensure that only authorized users (admins and customers) can access the appropriate parts of the system.
- Centralized Feedback Collection: The feedback submission process has been streamlined, allowing users to provide structured feedback, which is categorized for easier management.
- Real-Time Analytics: Interactive visualizations and metrics enable admins to view trends in feedback and engagement in real time.
- Interactive Dashboards: Dashboards are tailored to both customers for viewing their feedback history and admins for monitoring system-wide feedback data.
- Feedback Responses: Admins can respond directly to feedback entries, allowing for efficient communication and timely resolution of issues.
- Pending Feedback Tracking: The system continuously monitors and updates feedback that has not yet been addressed, ensuring no feedback is overlooked.

- Scalable and Secure: The system is designed to handle increased usage while maintaining strong security standards, including encrypted password storage and user activity tracking.

2. Methodology

2.1 Overview

Throughout the development, our project followed the Agile process model using the Scrum framework. The development was divided into five sprints, each lasting two weeks to enable fast delivery of functional components and regular feedback integration.

2.2 Key Scrum Practices

The following are some of the key Scrum practices we used during the course of our project:

- Sprint Planning Meetings: Defined sprint goals, prioritized backlog items, and assigned tasks.
- Daily Stand-ups: Short daily meetings to discuss progress, blockers, and plans.
- Sprint Reviews: Conducted team meetings to present completed work and collect feedback for improvement.
- Sprint Retrospectives: Reflected on the sprint to identify areas for improvement.

2.3 Version Control and Collaboration Tools

GitHub was used for version control and team collaboration throughout the project. Each team member created their own branches to work on specific features independently, which helped prevent conflicts and ensured smooth development. We adopted branching strategies to manage features, bug fixes, and releases efficiently. GitHub Issues and Pull Requests were actively used to track progress. Once a task was completed, we assigned one another as reviewers to ensure code quality before merging the approved changes into the main branch.

2.4 UML and Design Modeling

To ensure clear system understanding and communication among team members, several diagrams were created during the design phase which served as blueprints for system development and were updated as the project evolved.

- Use Case Diagram: Identified user interactions and system functionalities.
  (Refer to Appendix B)
- UML Class Diagram: Defined system structure, relationships, and responsibilities. (Refer to Appendix A)
- Sequence Diagram: Modeled the flow of interactions between objects during specific operations, helping to visualize the dynamic behavior of the system.
  (Refer to Appendix C)
- Documentation: Both inline code comments and external documentation (README file) were maintained.
- Testing: Unit tests were written using PHPUnit to verify individual modules.

2.5 Deployment Environment

To support development, testing, and demonstration, we deployed the system on Microsoft Azure Virtual Machines. This setup gave us a flexible and scalable environment, allowing the team to access the application remotely and test it under realistic deployment conditions. Azure’s infrastructure also supported our CI/CD pipeline integration and provided reliable uptime during sprint reviews.

3. Challenges Faced and Solutions

3.1 Overview

No software engineering project can be done smoothly. Almost all the time there would be a bit of a rough patch along the way and it would be up to the team to power through. This project was no exception. All the challenges are discussed in detail below.

3.2 Sprint Planning and Scope Creep

Sprint planning was crucial in setting deliverables for each iteration. Early on, we noticed that some tasks took longer than expected due to unforeseen complexity. In response, we enhanced our planning by involving the whole team in task estimation and introduced a structured process for handling change requests. These steps helped us stay agile while keeping sprint goals on track.

3.3 Communication Gaps in Daily Stand-Ups

Stand-up meetings, which were supposed to foster openness, quick issue resolution, and team coordination. But actually, there were some issues such as multitasking conversations, scheduling conflicts which often disrupted team coordination and delayed decision-making. To solve this, we adopted asynchronous communication tools, set clear agendas for meetings, and implemented a shared scheduling system to better accommodate everyone's availability and ensure smoother collaboration.

3.4 Version Control Conflicts

Collaboration was highly improved using GitHub, but it also posed its own issues. The most frequent problem was merge conflicts, especially when multiple contributors were handling overlapping parts of the codebase without syncing their branches on a regular basis. We addressed this by establishing clearer branching strategies, regularly pulling from the main branch, assigning each other as reviewers, and providing feedback before merging changes.

3.5 UML Modeling Limitations

Though UML diagrams were crucial in capturing the structure and behavior of the system, keeping these models updated throughout the development process was tedious. As the system matured, changes in logic or features necessitated frequent updates to use case, class, and sequence diagrams. To overcome these challenges, we streamlined the update process, improved communication between design and development teams, and explored more collaborative UML tools such as PlantUML to enhance efficiency.

3.6 Technical Debt and Code Integration

The modular development model encouraged reusability, but integration proved to have unexpected difficulties such as inconsistent coding styles, interface mismatches, and unplanned dependencies, which led to technical debt. We solved this by having team meetings to set clear coding standards, align on interface specifications, and manage module dependencies, which helped minimize integration issues and improve the development process.

3.7 Failures in CI/CD Pipeline

The implementation of continuous integration with the use of GitHub Actions was a huge step towards code stability but faced challenges such as defective configuration scripts and slow build times. These issues, including dependency installation errors and lengthy build processes, required deep CI tool knowledge to resolve, which delayed feature development. To resolve these, we refined the configuration scripts, optimized the build process, and leveraged team expertise to minimize disruptions and improve overall CI efficiency.

4. Key Lessons Learnt

4.1 Overview

The experience of creating the system using Agile and Scrum methodologies provided useful lessons in technical, collaboration, and project management aspects. These lessons will guide future software development and enhance our team's capacity to deliver effective and quality solutions.

4.2 Importance of Effective Sprint Planning

Of the most important lessons was the importance of realistic estimation in sprint planning. Underestimation or overestimation of the effort required by tasks can undermine team momentum and sprint goals. The team learned to break down tasks into smaller units, more wisely apply story points, and engage technical members in complex task estimation. This served to better align planning and execution.

4.3 Communication Needs To Be Intentional and Structured

Challenges faced in daily stand-ups reinforced the importance of clear, concise, and honest communication. The team learned that effective communication is not a matter of talking, it is a matter of communicating blockers earlier, listening carefully to others, and respecting the time box of each meeting. In the future, the team will maintain a daily agenda and change facilitators to improve participation and focus.

4.4 Version Control Discipline is Vital

The pains of merge conflicts taught the team the value of strict version control discipline. Adhering to a clearer branching strategy, syncing more often with the master branch, and writing informative commit messages became essential habits. Pre-merge inspections and enforcing code review conventions to minimize integration issues are what the team is going to do next.

4.5 Maintainable Documentation Saves Time

While it was laborious to maintain UML diagrams, it confirmed the adage that good documentation is worth it in the long run. If models and code were in sync, new joiners could easily onboard, and the team could more easily talk about system behavior. The team learned to update diagrams incrementally rather than make documentation a one-time activity.

4.6 Code Consistency and Collaboration Go Hand-in-Hand

Integration of the modular components made a case for the consistent coding style and shared design decisions. The group plans to adopt a shared coding style guide and hold frequent code walkthroughs and pair programming on key areas of the code. This will make the code more integrated and easier to maintain.

4.7 Testing is a Core Part of Development, Not an Afterthought

Partial test coverage showed the risks of optional testing. The team learned that robust unit and integration testing ensures code reliability in the long term. Tests created alongside development—not afterwards—will be a new best practice. Additionally, adding test automation to the CI pipeline will identify issues early and reduce regression.

5. User Credentials and Access Levels

When a new user is registered, they are automatically assigned the role of a Customer.

To test the Admin side, use the following credentials:

- **Username:** utosun@gmail.com
- **Password:** UmutTosun@123

This allows you to access and manage admin-specific features.

6. Conclusion

The achievement of the development and implementation of DataSphere is a significant milestone in the long-standing problem of customer feedback fragmentation for African businesses. Through the removal of inefficiencies in the guise of sluggish response times, disparate data gathering, and decentralized feedback streams, the project is an example of how a technology-supported structured solution can transform the manner in which businesses interact with and respond to customers. DataSphere's user-friendly interface and real-time analysis allow companies to make informed decisions, quickly address customer concerns, and create a responsiveness and continuous improvement culture.

One of the secrets to launching this project was using the Agile process model with the Scrum framework, which provided a nimble and team-oriented space in which to build. Sprints, regular stand-ups, and retrospectives kept the team aligned, responsive to change, and generating incremental value throughout the entire project. By using modern development technology such as version control via GitHub and system architecture via UML modeling, the team stayed transparent, structured, and aligned throughout all phases of the project.

But the process was not smooth sailing. From merge conflict resolution to workload allocation, from scope shift to performance optimization, the project tested the team's endurance and problem-solving skills. Stressful as they were, they contributed significantly to our professional and technical growth. They reinforced the importance of clear communication, prudent planning, and adaptability—qualities essential not only in software development, but in any cooperative project.

Achievement of the aforementioned goals ranging from centralized collection of feedback to safe authentication all the way to responsive dashboards and data visualizations steeped in knowledge—mirrors the passion of the team to create a robust, scalable, and accessible platform. DataSphere is not just a feedback collection platform; it is a business strategy for organizations looking to evolve in response to customer expectations in an increasingly digitising economy.

In the future, the project is full of opportunities for improvement and growth. Support for AI-powered sentiment analysis, multilingual capability, and API integration with established CRM software will further enhance its utility. Recruiting more varied businesses and receiving feedback from diverse industries will allow for the further development of the system and increase its adaptability.

In summary, this project illustrates the strength of technology and collaboration in addressing genuine issues in real life. DataSphere is an effective and actionable solution that allows African companies to bridge the difference between customer experience and service. It is our expectation that the platform will still grow and provoke similar innovations to enhance customer-firm relationships across the continent and the world.

7. References

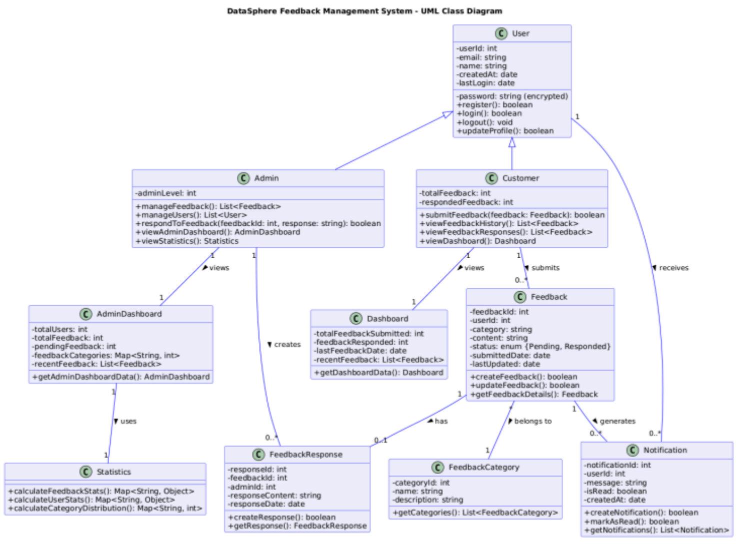
Booysen, M. (2025, February 5). *Customer Service Excellence: The Importance of Feedback Loops in Customer Service Processes.* Lionesses of Africa. https://www.lionessesofafrica.com/blog/2025/02/05/customer-service-excellence-the-importance-of-feedback-loops-in-customer-service-processes

Hinson, R. E., Adeola, O., Lituchy, T. R., & Amartey, A. F. O. (2020). *Customer Service Management in Africa: A Strategic and Operational Perspective.* Taylor and Francis. https://doi.org/10.4324/9780429031342

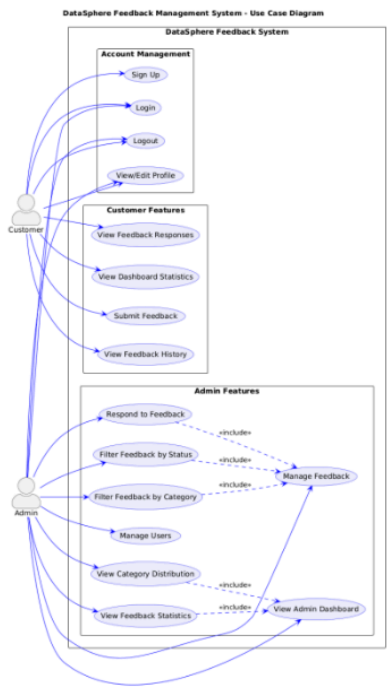
Real-Time feedback is still a major problem for most companies. (2023, January 24). CRM Magazine. https://www.destinationcrm.com/Articles/ReadArticle.aspx?ArticleID=156808

Nyakey, V. Y. (2024, December 14). *Improving Customer Service in Africa: Strategies for Success*. ModernGhana.

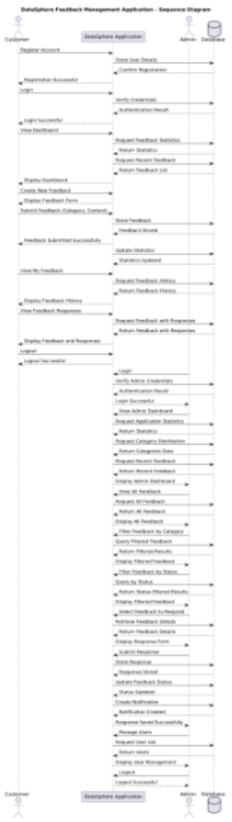
8. Appendices


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>
views
creates
uses
views
submits
receives
has
belongs to
generates
1
0..*
0..1

Appendix A: UML Class Diagram
[[Click Here to view the UML Class Diagram]]

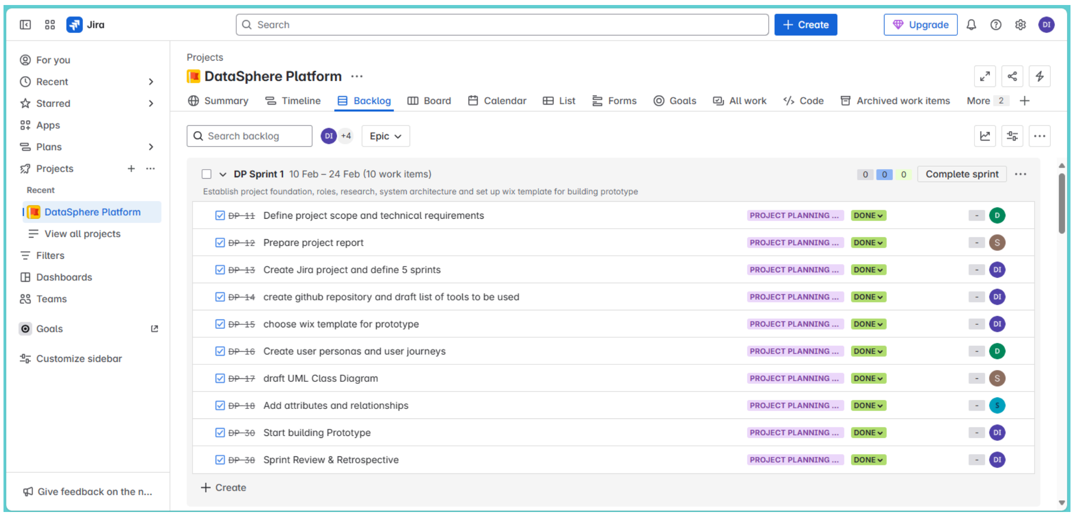


Appendix B: Use Case Diagram
[[Click Here to view the Use Case Diagram]]



Appendix C: Sequence Diagram

[Click Here to view the Sequence Diagram]



Projects

DataSphere Platform

DP Sprint 1 10 Feb – 24 Feb (10 work items) 0 0 0

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

Appendix D: DataSphere Platform - Backlog - Jira

[Click here to view the DataSphere Platform - Backlog - Jira]