PROJECT TITLE : Courier Services Management System

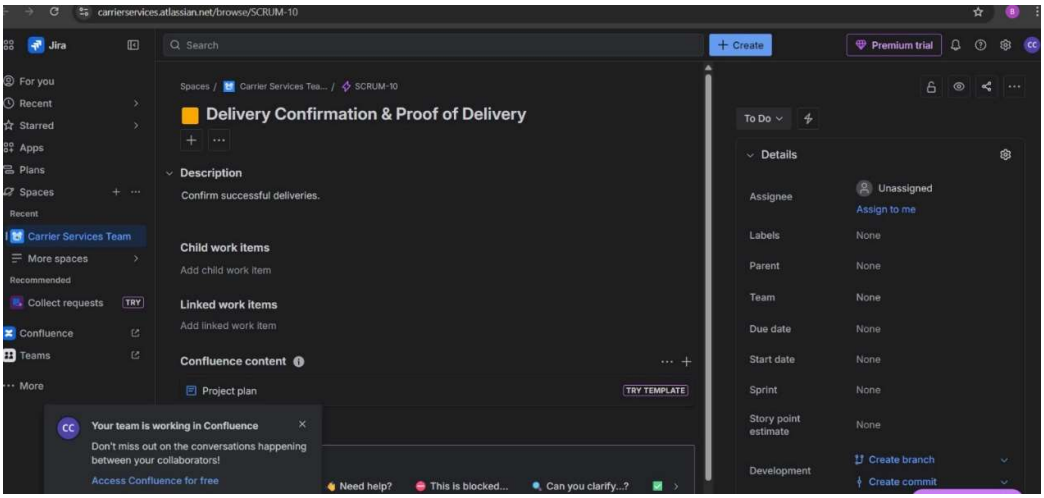
GROUP MEMBERS;

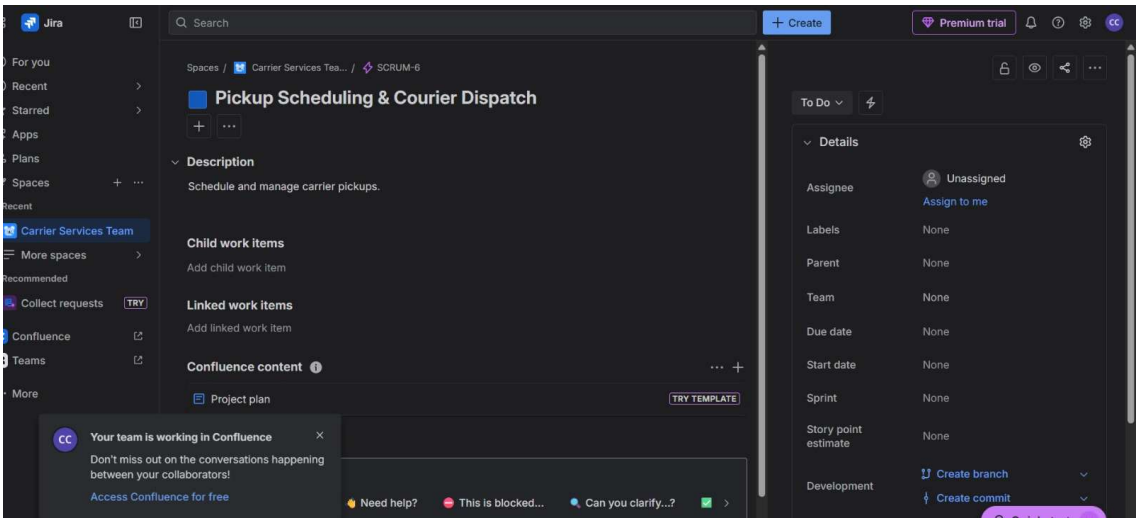
NAME	REG NUMBER
1.Brian Gitau	E020-01-0894/2023
2. Vivian Wanjiku	E020-01-0957/2023
3.Alex Miano	E020-01-0938/2023
4. Eldad Wahome	E020-01-0885/2023
5. Waithaka Wakahiu	E020-01-0887/2023
6. Cynthia Cherono	E020-01-0919/2023
7. Dennis Kipchumba	E020-01-0925/2023
8. James Thuo	E020-01-0956/2023
9. Benson Karobia	E020-01-0951/2023
10.Dorine Chepngeno	E020-01-0921/2023

Description:

The Courier Services Management System is designed to streamline parcel delivery operations by managing senders, receivers, station admins, and the general manager. The system ensures efficient parcel tracking, user registration, and role-based management. It supports data handling for both senders and receivers, while enabling administrators to monitor station activities and managers to oversee overall operations. Key Stakeholders & Roles: • General Manager – Oversees all courier operations, generates reports, and manages system-wide activities. • Station Admin – Manages parcels at a specific station, updates parcel status, and verifies sender/receiver details. • Sender (User) – Provides parcel details including Name, ID, Parcel ID, and location. • Receiver (User) – Provides Name, Phone number, ID, Parcel ID, and receiver location.

SECTION 1:



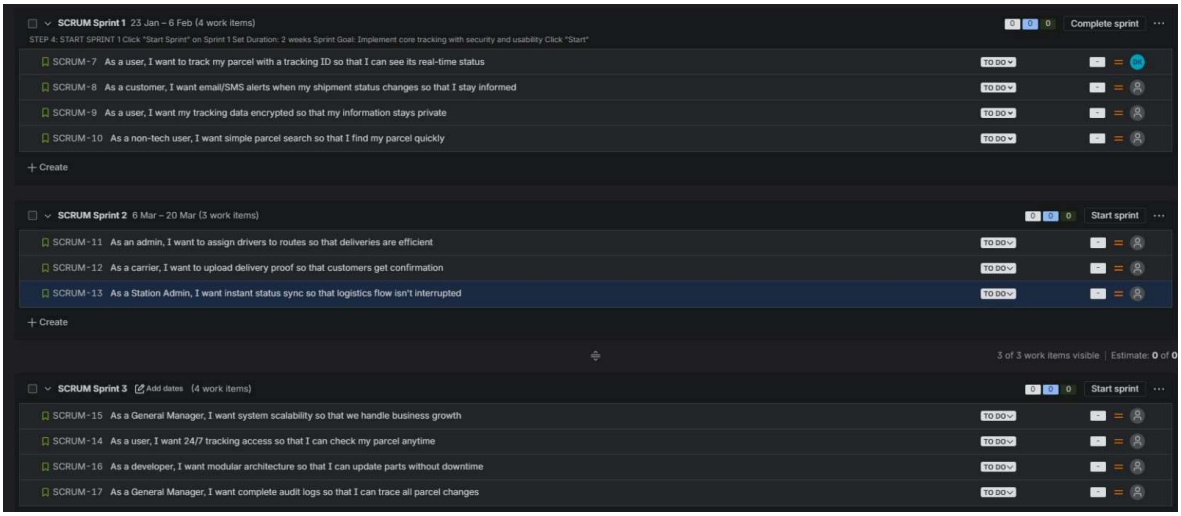


Specific epics within the "Carrier Services Team" space in Jira:

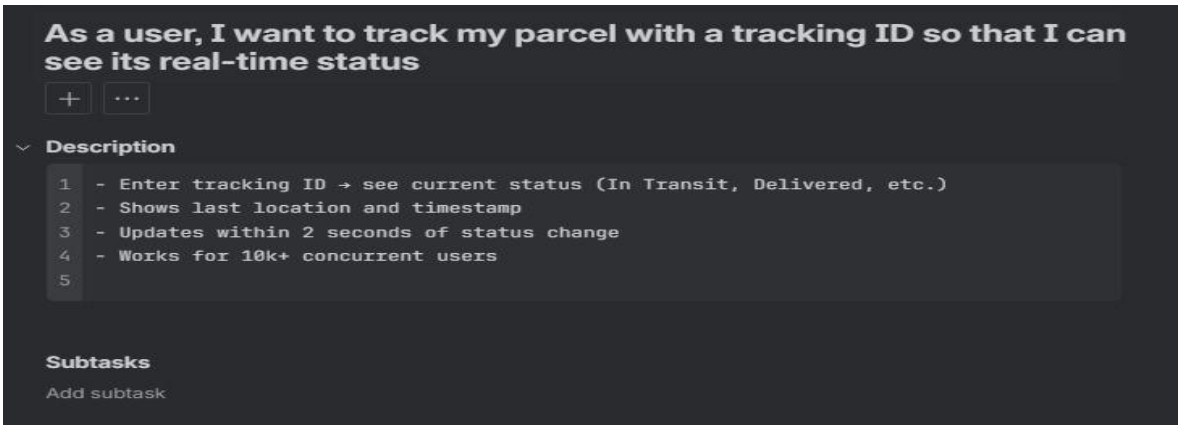
- **Delivery Confirmation & Proof of Delivery (SCRUM-10):** This epic is focused on the functional requirements for confirming successful deliveries.
- **Pickup Scheduling & Courier Dispatch (SCRUM-6):** This epic covers the management and scheduling of carrier pickups.

SECTION 2:

JIRA SPRINT



USER STORIES



As a customer, I want email/SMS alerts when my shipment status changes so that I stay informed

Description

```
 System sends email within 1 minute of status change
- System sends SMS within 1 minute of status change
- User selects preference (email/SMS/both)
- 99.9% notification delivery rate

```

Subtasks

As a user, I want my tracking data encrypted so that my information stays private

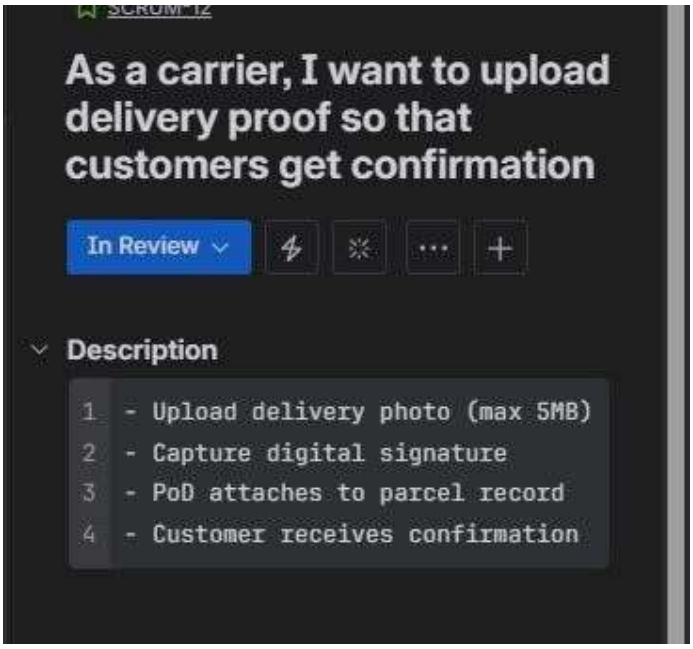
Description

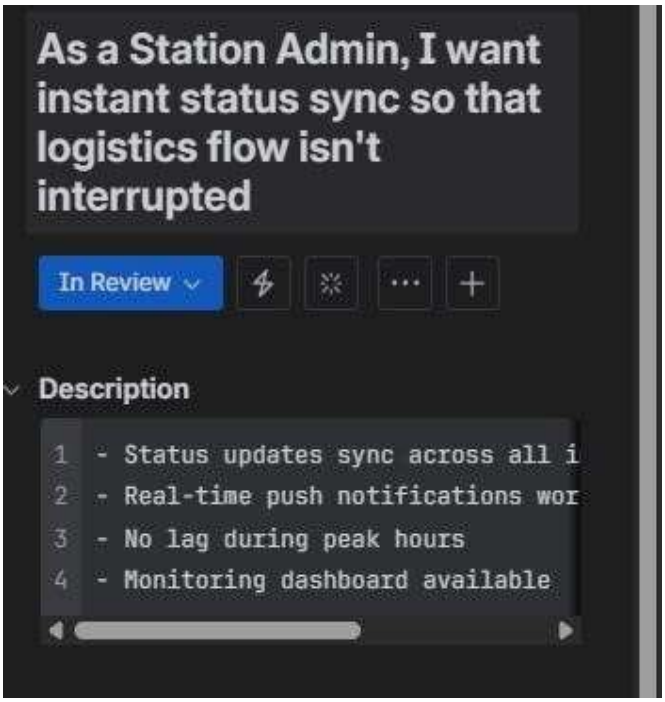
```
- PII encrypted with AES-256 in database
- TLS 1.3 for all data transmission
- Encrypted tracking history logs
- Security audit passes
```

As a non-tech user, I want simple parcel search so that I find my parcel quickly

Description

```
- Novice user finds parcel in < 15 seconds
- Simple, clean search interface
- Clear error messages
- Mobile-responsive design
```


SCRUM-12
As a carrier, I want to upload delivery proof so that customers get confirmation
In Review
Description
1 - Upload delivery photo (max 5MB)
2 - Capture digital signature
3 - PoD attaches to parcel record
4 - Customer receives confirmation


As a Station Admin, I want instant status sync so that logistics flow isn't interrupted
In Review
Description
1 - Status updates sync across all i
2 - Real-time push notifications wor
3 - No lag during peak hours
4 - Monitoring dashboard available

As a General Manager, I want complete audit logs so that I can trace all parcel changes

Description

```
- Log every parcel status change
- Record Admin ID, Vehicle ID, timestamp
- Immutable log storage (7-year retention)
- Generate audit reports
```

Subtasks

As a developer, I want modular architecture so that I can update parts without downtime

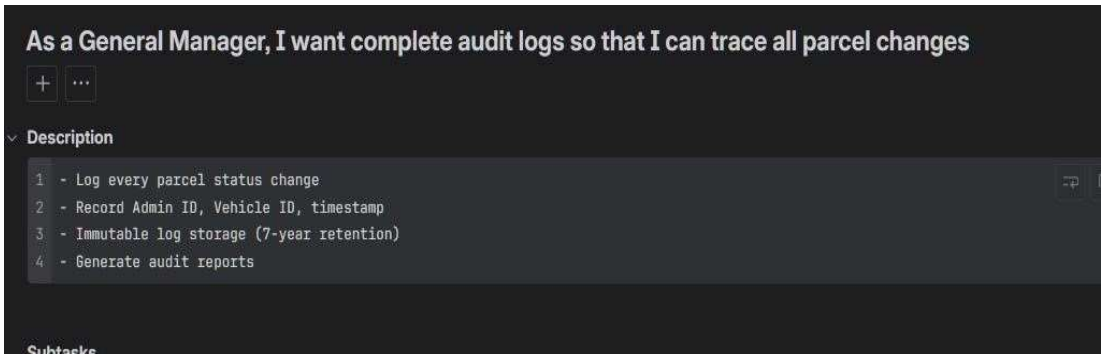
Description

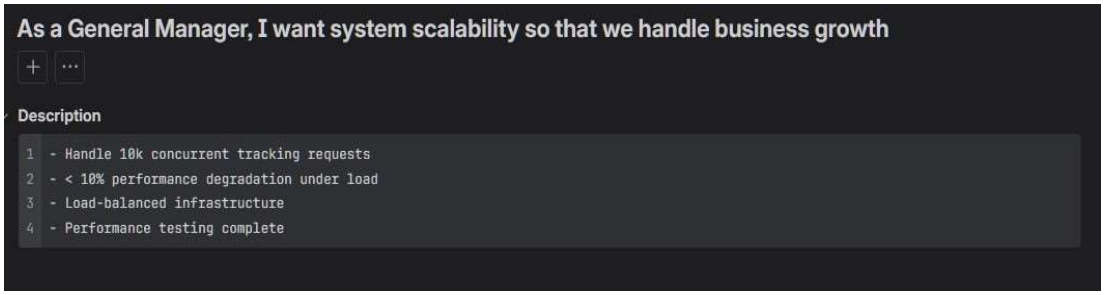
```
- Update Admin module without affecting tracking
- Zero-downtime deployments
- Microservices communicate via APIs
- Independent service scaling

```

As a General Manager, I want system scalability so that we handle business growth

Description

```
- Handle 10k concurrent tracking requests
- < 10% performance degradation under load
- Load-balanced infrastructure
- Performance testing complete
```





SECTION 3: ACCEPTANCE CRITERIA

Functional User Stories

- **Parcel Tracking**:
  - Users can enter a tracking ID to view real-time status (e.g., In Transit, Delivered).
  - Display includes the last known location and a timestamp.
  - Status updates refresh within 2 seconds of a system change.
  - System maintains performance for 10k+ concurrent users.
- **Shipment Notifications**:
  - Automated email alerts are sent within 1 minute of a status change.
  - Automated SMS alerts are sent within 1 minute of a status change.
  - Users can choose notification preferences (Email, SMS, or Both).
  - The system ensures a 99.9% notification delivery rate.
- **User Interface & Accessibility**:
  - Search functionality allows novice users to find parcels in under 15 seconds.

    - Interface is designed to be simple, clean, and intuitive.
    - System provides clear, actionable error messages.
    - The design is fully mobile-responsive.
- **Driver & Route Management**:
    - Admins can manually assign drivers to specific delivery routes.
    - System supports auto-assignment based on real-time driver availability.
    - Admins have access to a driver workload dashboard.
    - The module supports 500+ simultaneous assignments.
- **Delivery Confirmation**:
    - Carriers can upload delivery photos (max 5MB) and capture digital signatures.
    - Proof of Delivery (PoD) is automatically attached to the parcel record.
    - Customers receive an immediate confirmation upon delivery.
- **Logistics Synchronization**:
    - Status updates sync instantly across all system instances.
    - Real-time push notifications are active for station administrators.
    - System architecture ensures no lag during peak operational hours.
    - A live monitoring dashboard is available for logistics oversight.

Security & Technical User Stories

- **Data Security**:
    - Personally Identifiable Information (PII) is encrypted using AES-256 at rest.
    - All data in transit is protected via TLS 1.3.
    - Tracking history logs are stored in an encrypted format.
    - The system must pass all formal security audits.
- **Audit & Compliance**:
    - Every parcel status change is logged automatically.
    - Logs must capture Admin ID, Vehicle ID, and a precise timestamp.
    - Log storage is immutable with a 7-year retention policy.
    - The system provides a function to generate comprehensive audit reports.
- **Scalability**:
    - System handles 10k concurrent tracking requests efficiently.
    - Performance degradation remains under 15% during peak load.
    - Infrastructure utilizes load-balancing to distribute traffic.
    - Full performance testing is mandatory for deployment.
- **Modular Architecture**:
    - The Admin module can be updated without impacting tracking services.
    - The system supports zero-downtime deployment cycles.
    - Microservices communicate exclusively via standardized APIs.
    - Each service is capable of independent scaling based on demand.

SECTION 4: Team assignment and workflow.

Individual Team Member Assignments.

👥 Work Division for 10 Members

Name	Role	Responsibility	Deliverable
1.Brian Gitau	GitHub Lead	Create repository, setup README, documentation branch, ensure commits	Public repo link + README.md
2. Vivian Wanjiku	Documentation Manager	Compile group members list, maintain structure, merge deliverables	Organized repo with docs
3.Alex Miano	Requirements Analyst (Functional)	Draft 7-10 functional requirements (FR-01..FR-10)	SRS functional requirements section
4. Eldad Wahome	Requirements Analyst (Non-Functional)	Draft 5-7 non-functional requirements	SRS non-functional requirements section
5. Waithaka Wakahiu	System Overview Writer	Write system purpose, scope, stakeholders, modules	SRS overview section
6. Cynthia Cherono	Jira Project Manager	Setup Jira Scrum project, create Epics	Jira backlog screenshot
7. Dennis Kipchumba	Jira Sprint Planner	Convert requirements into user stories, assign story points, plan Sprint 1	Jira sprint screenshots
8. James Thuo	UML Designer (Use Case & Activity)	Create Use Case Diagram + Activity Diagram	UML diagrams PDF
9. Benson Karobia	UML Designer (Sequence & Class)	Create Sequence Diagram + Class Diagram	UML diagrams PDF
10.Dorine Chepngeno	QA & Integration Lead	Review all deliverables, ensure consistency, finalize PDF reports	Final submission in GitHub

Each team member was assigned specific responsibilities based on their role to ensure effective collaboration and accountability throughout the project.

1. Brian Gitau – GitHub Lead

Brian Gitau was responsible for setting up the GitHub repository, creating the README file, and managing the documentation branch. He ensured proper version control, consistent commit history, and smooth integration of all team contributions.

Deliverable: Public GitHub repository link and README.md file.

2. Vivian Wanjiku – Documentation Manager

Vivian Wanjiku compiled the list of group members, maintained document structure, and merged all deliverables into a well-organized repository. She ensured documentation consistency and clarity across all project files.

Deliverable: Organized repository with complete documentation.

👥 Work Division for 10 Members

Name	Role	Responsibility	Deliverable
LBrian Gitau	GitHub Lead	Create repository, setup README, documentation branch, ensure commits	Public repo link + README.md
2. Vivian Wanjiku	Documentation Manager	Compile group members list, maintain structure, merge deliverables	Organized repo with docs
3.Alex Miano	Requirements Analyst (Functional)	Draft 7–10 functional requirements (FR-01 – FR-10)	SRS functional requirements section
4. Eldad Wahome	Requirements Analyst (Non-Functional)	Draft 5–7 non-functional requirements	SRS non-functional requirements section
5. Waithaka Wakahiu	System Overview Writer	Write system purpose, scope, stakeholders, modules	SRS overview section
6. Cynthia Cherono	Jira Project Manager	Setup Jira Scrum project, create Epics	Jira backlog screenshot
7. Dennis Kipchumba	Jira Sprint Planner	Convert requirements into user stories, assign story points, plan Sprint 1	Jira sprint screenshots
8. James Thuo	UML Designer (Use Case & Activity)	Create Use Case Diagram + Activity Diagram	UML diagrams PDF
9. Benson Karobia	UML Designer (Sequence & Class)	Create Sequence Diagram + Class Diagram	UML diagrams PDF
10.Dorine Chepngeno	QA & Integration Lead	Review all deliverables, ensure consistency, finalize PDF reports	Final submission in GitHub

3. Alex Miano – Requirements Analyst (Functional)

Alex Miano served as the Requirements Analyst (Functional) for the logistics and courier services project. His primary responsibility was to define the core functional capabilities of the system by specifying what the software must do to support business operations.

He drafted a structured list of 7–10 functional requirements, clearly labeled from FR-01 to FR-10, covering key system functionalities. These requirements formed the Functional Requirements section of the Software Requirements Specification (SRS) and provided a foundation for user story creation and system design.

Alex documented all functional requirements using Markdown to ensure clarity, consistency, and version control compatibility. He also actively contributed to the project repository by making at least two descriptive GitHub commits, reflecting the progression and refinement of the functional requirements.

Deliverable: SRS Functional Requirements section (FR-01 to FR-10) documented in Markdown and committed to GitHub.

FR-05	Real-Time Tracking	The system shall provide a tracking interface where users can enter a tracking ID to see the current status (e.g., In Transit, Out for Delivery).
FR-06	Automated Notifications	The system shall send email or SMS alerts to the customer whenever the shipment status changes.
FR-07	Courier Assignment	The system shall allow administrators to manually or automatically assign specific local drivers to "Last-Mile" delivery routes.
FR-08	Proof of Delivery (PoD)	The system shall allow the carrier to upload a digital signature or photo as proof that the package was successfully delivered.

Requirement ID	Feature Name	Description
FR-01	User Registration	The system shall allow shippers, carriers, and administrators to create and manage secure accounts with role-based access.
FR-02	Carrier Rate Comparison	The system shall fetch and display real-time shipping rates from different carriers (e.g., DHL, FedEx) based on package weight and destination.
FR-03	Shipment Booking	The system shall allow users to select a carrier and generate a unique tracking ID for a new shipment.
FR-04	Label Generation	The system shall generate a downloadable PDF shipping label containing the barcode, origin, and destination address.

4. Eldad Wahome – Requirements Analyst (Non-Functional)

Eldad Wahome focused on defining non-functional requirements such as performance, security, reliability, and usability. These requirements ensured the system met quality and operational standards.

Deliverable: Non-functional requirements section of the SRS document.

```
SRS Section: Non-Functional Requirements
1. Performance (Latency)
The system must provide real-time updates to ensure logistics flow is not interrupted.

Requirement: The system shall process and reflect "Station Admin" updates (Vehicle ID and Time) across all user interfaces
within 2 seconds of the data being committed to the database.

2. Scalability (Throughput)
The platform must handle growth in shipment volume and user traffic.

Requirement: The system architecture must support a minimum of 10,000 concurrent tracking requests and 500 simultaneous
admin entries without a degradation in response time exceeding 10%.

3. Security (Data Protection)
Protection of sensitive user attributes is a legal and operational necessity.

Requirement: All PII (Personally Identifiable Information), including Sender/Receiver Names, Phone numbers, and IDs, must
be encrypted using AES-256 standards both at rest and in transit (via TLS 1.3).

4. Availability (Uptime)
Logistics services operate around the clock; downtime results in physical shipment backlogs.

Requirement: The system shall maintain an availability of 99.9% (3-nines), ensuring that Senders and Receivers can access
the "Parcel ID" lookup service 24/7.

5. Usability (Accessibility)
The interface must be accessible to users with varying levels of technical literacy.

Requirement: The user interface for Senders and Receivers must achieve a System Usability Scale (SUS) score of 80 or
higher, ensuring that a Parcel ID search can be completed in under 15 seconds by a novice user.

6. Maintainability (Serviceability)
The system must be built so that developers can fix issues quickly without disrupting the whole chain.

Requirement: The system shall be developed using a modular or microservices architecture, allowing the "Station Admin"
module to be updated or patched without taking the "Sender/Receiver" tracking portal offline.

7. Auditability (Traceability)
For the General Manager, knowing "who did what" is critical for resolving lost parcel disputes.

Requirement: The system must maintain an immutable audit log for every change made to a Parcel ID status, recording the
Station Admin ID, the Vehicle ID used, and the exact timestamp of the modification.
ID      Category        Primary Stakeholder     Metric
NFR-01  Performance     Station Admin   < 2s Sync Time
NFR-02  Scalability     General Manager 10k Concurrent Users
NFR-03  Security        Users (Sender/Receiver) AES-256 Encryption
NFR-04  Availability    All Users       99.9% Uptime
NFR-05  Usability       Users (Sender/Receiver) < 15s to find Parcel
NFR-06  Maintainability Developers/IT   Zero-downtime patches
NFR-07  Auditability    General Manager 100% Traceability
```

5. Waithaka Wakahiu – System Overview Writer

Waithaka Wakahiu was responsible for documenting the System Overview section of the Software Requirements Specification (SRS). His role focused on clearly defining the system's purpose, scope, stakeholders, core modules, and functional workflow to provide a high-level understanding of the Courier Services Management System.

He documented the system purpose, explaining how the system automates and centralizes parcel delivery operations, reduces data redundancy, minimizes human error, and enforces secure role-based access control. The system scope was clearly defined to cover the complete parcel lifecycle from registration to delivery verification, while explicitly excluding external third-party integrations and payment processing.

He also identified and described all key stakeholders, including the General Manager, Station Admin, Sender, and Receiver, outlining their roles and interactions with the system. In addition, he detailed the major system modules, such as authentication, parcel management, user information management, and reporting and analytics.

Finally, he provided a functional overview describing the trigger-based workflow covering registration, validation, tracking, completion, and auditing.

Deliverable: System Overview and Functional Overview sections of the SRS document.

6**. Cynthia Cherono – Jira Project Manager**

Cynthia Cherono set up the Jira Scrum project and created epics that represented the major system functionalities. She ensured proper backlog organization and alignment with project requirements.

Deliverable: Jira backlog with epics (screenshot).

7. Dennis Kipchumba – Jira Sprint Planner

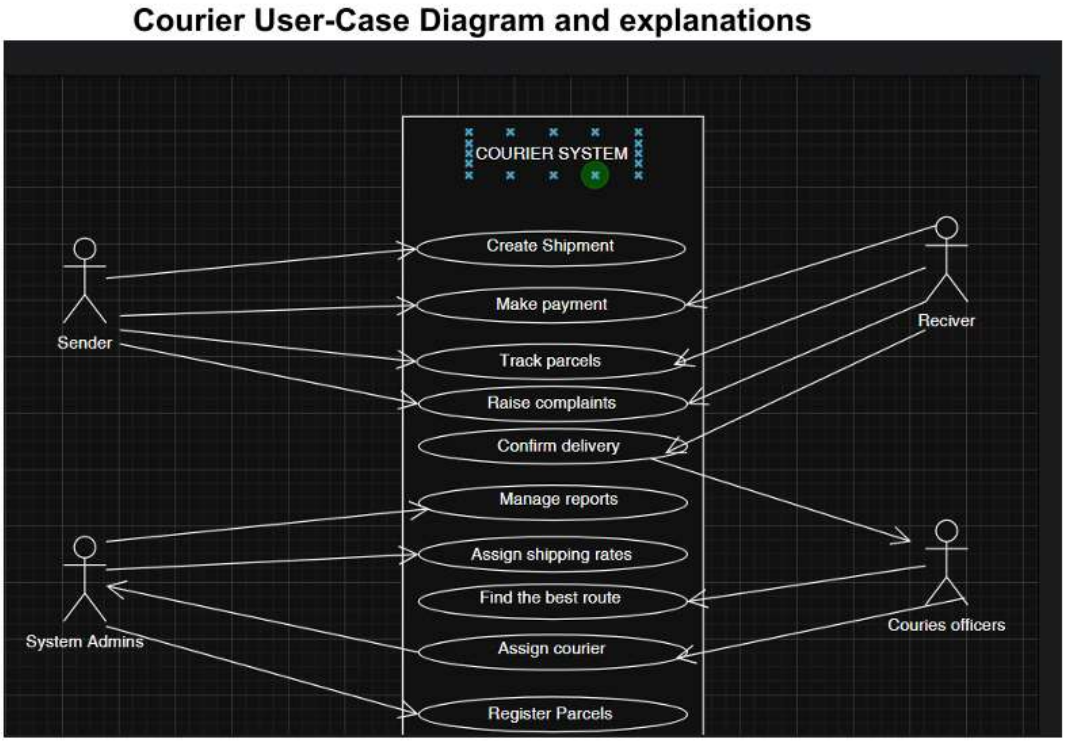
Dennis Kipchumba converted requirements into user stories, assigned story points, and planned Sprint 1. He ensured sprint goals aligned with project objectives and team capacity.

Deliverable: Jira active sprint screenshots.

8. James Thuo – UML Designer (Use Case & Activity)

James Thuo was responsible for designing the Use Case Diagram and Activity Diagram for the Courier Services Management System. His role focused on visually representing the system's functional behavior and workflows to guide development and ensure clarity for all stakeholders.

Deliverable: UML Use Case and Activity Diagrams submitted in PDF format and committed to the GitHub repository.



Courier System – Use Case UML Explanation

1.Sender Use Case Relationships

Create Shipment – Sender initiates shipment by providing parcel, sender, and receiver details.

Make Payment – Sender pays for shipment services. Payment is mandatory for shipment processing.
Track Parcels – Sender monitors parcel location and delivery status.

Raise Complaints – Sender reports issues such as delays, loss, or poor service.

Confirm Delivery – Sender confirms successful delivery where applicable.

2.Receiver Use Case Relationships

Track Parcels – Receiver checks parcel movement and expected delivery time.

Raise Complaints – Receiver reports damaged, delayed, or missing parcels.

Confirm Delivery – Receiver confirms receipt of the parcel.

3.System Admin Use Case Relationships

Manage Reports – Admin generates and reviews system, delivery, and financial reports.

Assign Shipping Rates – Admin sets and updates shipping costs based on distance, weight, or service type.

Assign Courier – Admin assigns shipments to available courier officers.

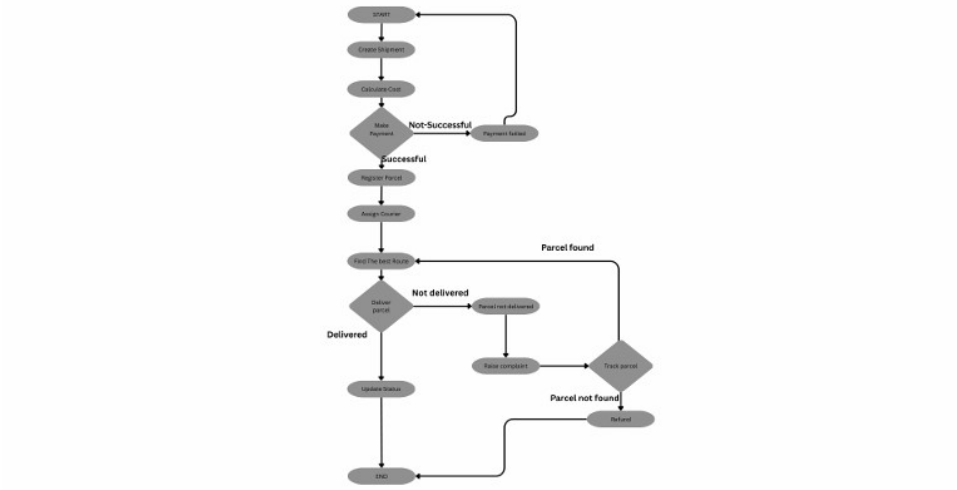
Register Parcels – Admin registers parcels manually when required.

4.Courier Officers Use Case Relationships

Find the Best Route – Courier officers determine optimal delivery routes.

Assign Courier – Courier officers receive and accept assigned deliveries.

Confirm Delivery – Courier officers update the system after successful delivery.



Start: Begins with creating a shipment and calculating costs.
Payment: User makes payment; if unsuccessful, process ends with "Payment Failed." If successful, parcel is registered, courier assigned, and best route found (assuming parcel is located).
Delivery: Parcel is delivered; if successful, status updates and process ends. If not delivered, user can raise a complaint and track the parcel.
Failure Paths: If parcel not found during tracking, leads to refund.
End: Concludes after status update or refund.

9. Benson Karobia – UML Designer (Sequence & Class)

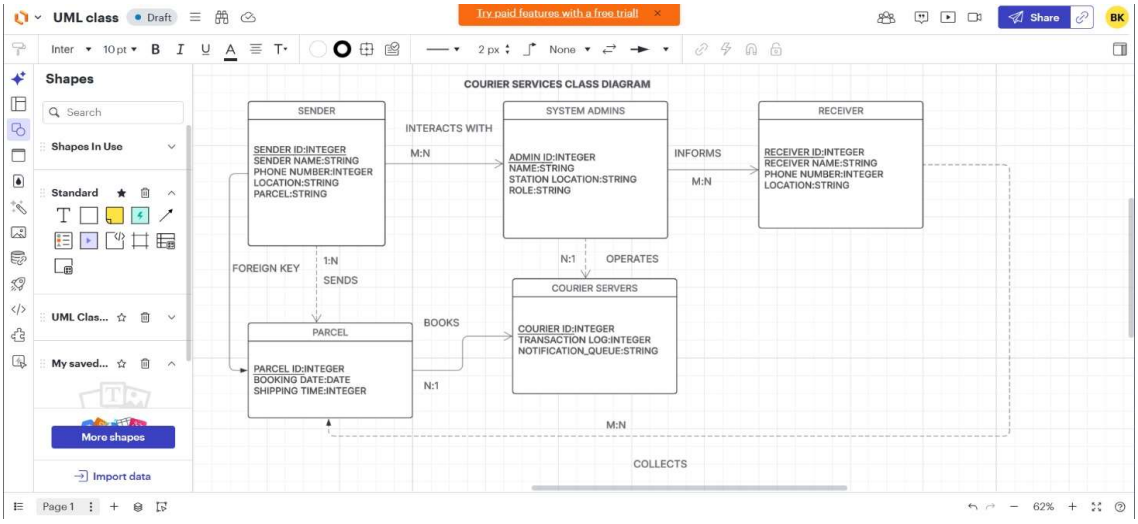
Benson Karobia was responsible for designing the Sequence Diagram and Class Diagram for the Courier Services Management System. His role focused on modeling the system's internal interactions and structural architecture based on the functional requirements defined in the SRS.

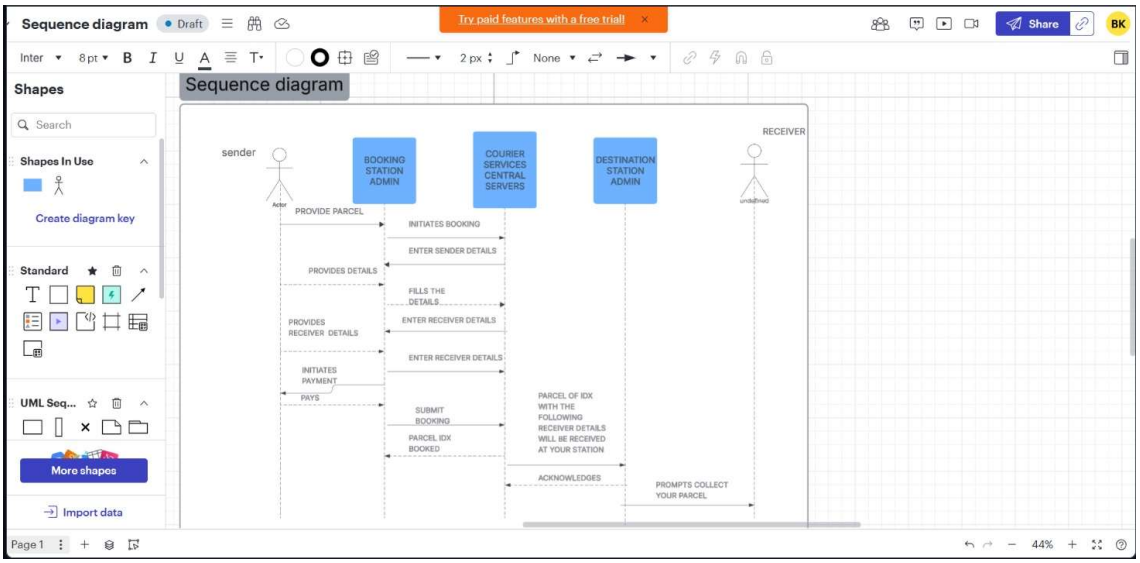
He developed the Sequence Diagram to illustrate the chronological interaction between system actors and system components during key processes such as parcel registration, status updates, and delivery confirmation. This diagram helped clarify message flow, object interaction, and system behavior over time.

In addition, Benson designed the Class Diagram to represent the system's static structure, including classes, attributes, methods, and relationships such as association and inheritance. This diagram provided a clear blueprint for understanding data organization and system architecture.

All UML diagrams were created following standard UML conventions and were aligned with the system requirements to ensure consistency and accuracy.

Deliverable: UML Sequence Diagram and Class Diagram submitted in PDF format and committed to the GitHub repository.





10. Dorine Chepngeno – QA & Integration Lead

Dorine Chepngeno served as the Quality Assurance (QA) and Integration Lead for the Courier Services Management System project. Her primary responsibility was to review all project deliverables to ensure accuracy, consistency, and compliance with the project requirements and naming conventions.

She verified that all documentation, UML diagrams, and Jira artifacts aligned with the Software Requirements Specification (SRS) and the defined project scope. Dorine also ensured that completed tasks met their acceptance criteria before being marked as Done in Jira.

In addition, she integrated individual contributions into a single, coherent final PDF report and confirmed that all required files were properly committed and organized in the GitHub repository.

Deliverable: Final integrated PDF report and verified GitHub submission.

Workflow Management and Issue Progression

(Screenshot: Jira board showing To Do → In Progress → Done)

Each task followed a defined workflow in Jira:

Tasks were created and placed in To Do

Moved to In Progress when actively worked on

Marked Done after completion and review