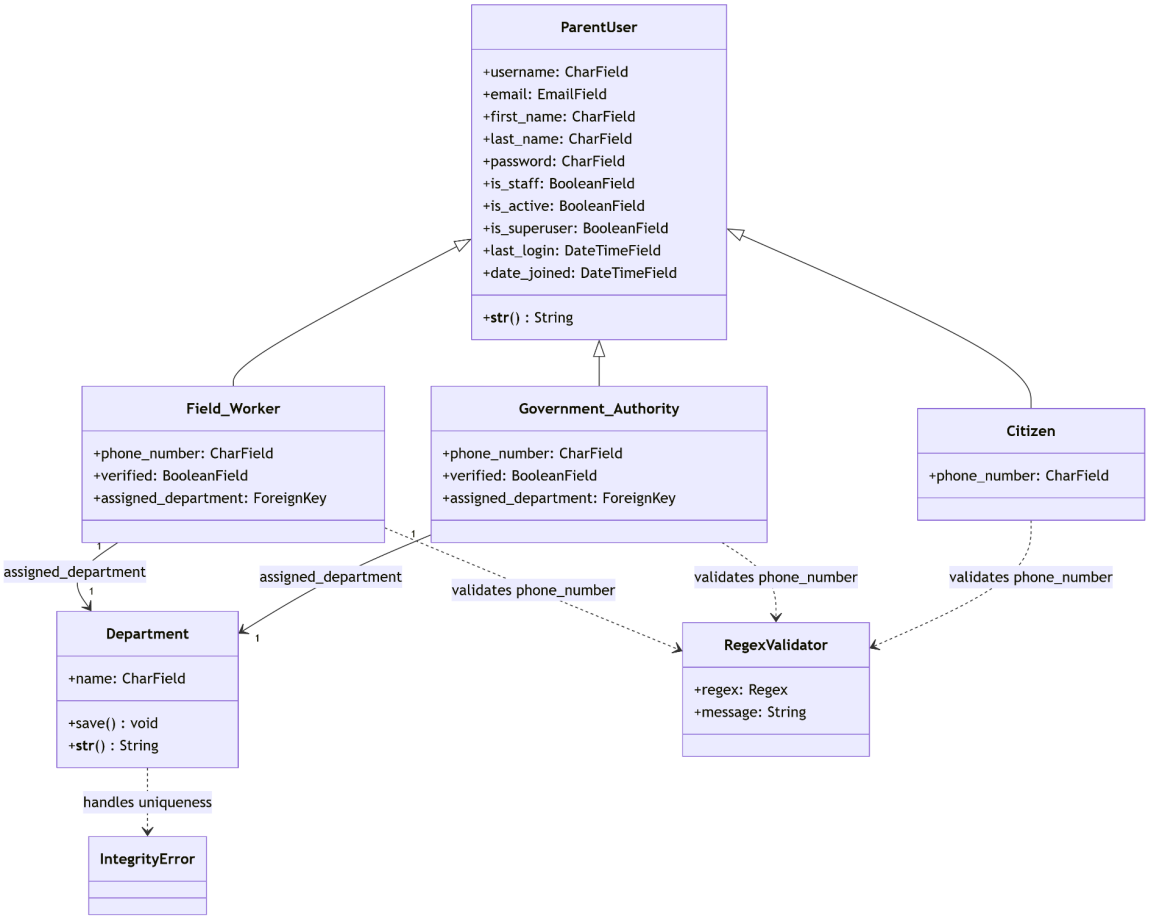
Sprint 1: Project Setup & User Authentication

During the first sprint, the foundation of the application was established.
The focus was on setting up a secure and reliable authentication mechanism for all user types in the system - **Citizens**, **Government Authorities**, and **Field Workers**.

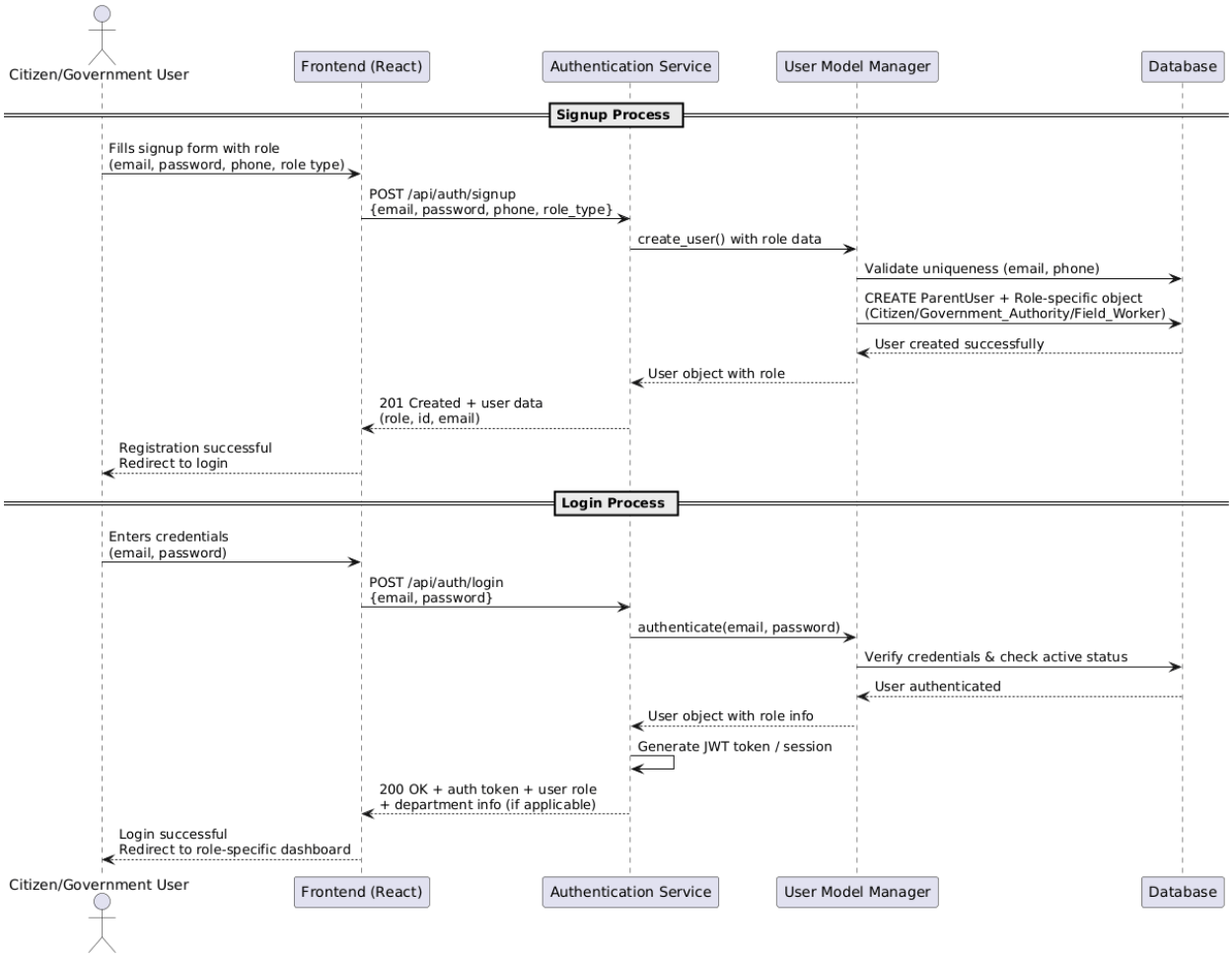
Key tasks included:

- Implementing **Login and Sign-Up** workflows tailored to different user roles.
- Integrating **JWT/Session-based authentication** to ensure safe and authenticated access to resources.
- Setting up an **Email Verification system** to validate user identities and prevent unauthorized registrations.

This sprint laid the groundwork for all future modules by ensuring the system had a robust, secure entry point.
Class Diagram:



Sequence Diagram:



Note: User Model Manager is not explicitly defined in our code as Django automatically creates a default manager called `objects` when you do: User.objects.create()

Sprint 2: Citizen Complaint Submission & Public Feed

The second sprint introduced the core feature of the platform - **citizen complaint reporting**.

Major components delivered:

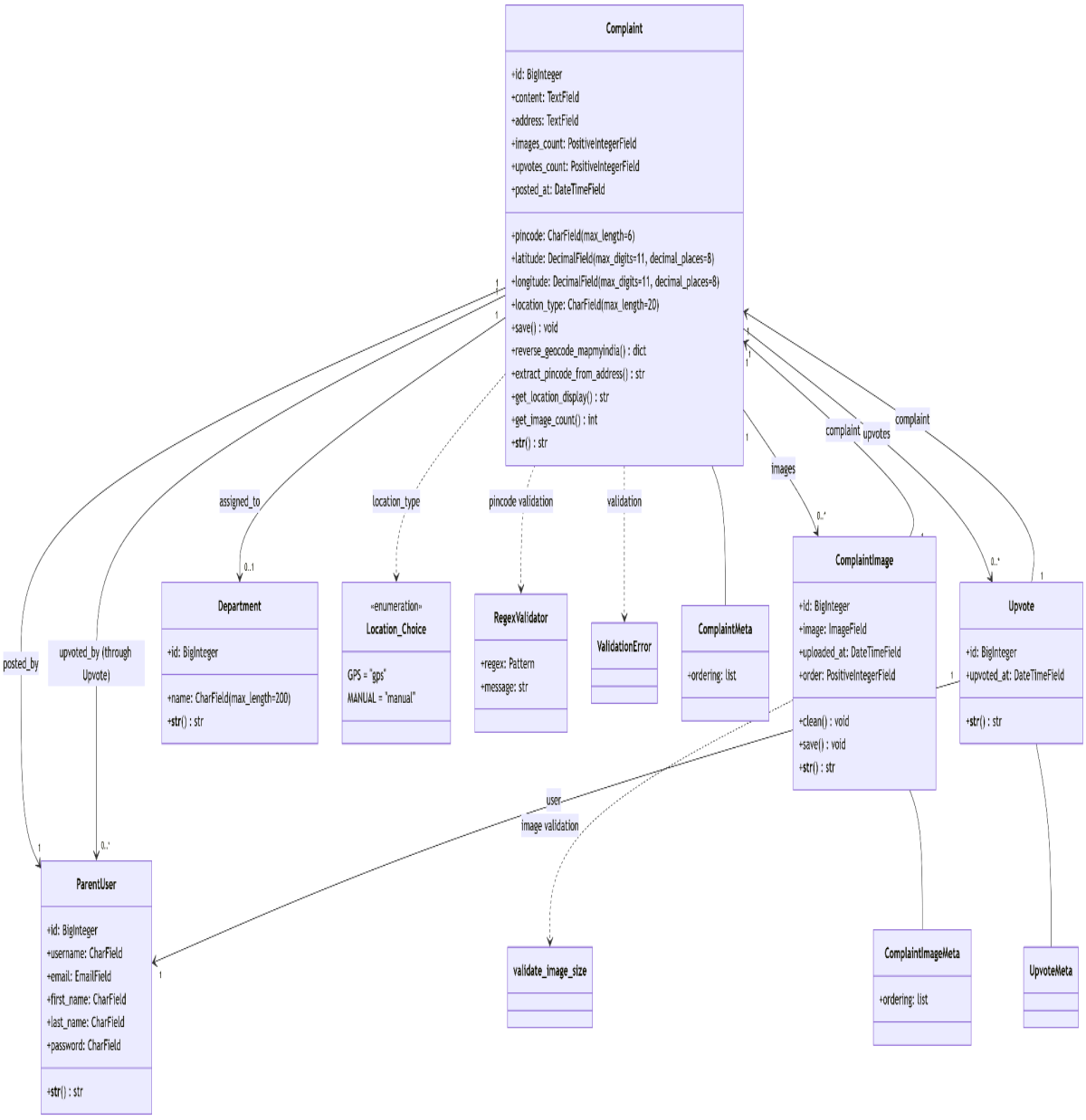
- A comprehensive **complaint submission form** supporting:
    - Issue description
    - Category selection through dropdown

    - Location selection via **auto-detect (GPS)** or **manual input**
    - Uploading up to **four images** for better visualization

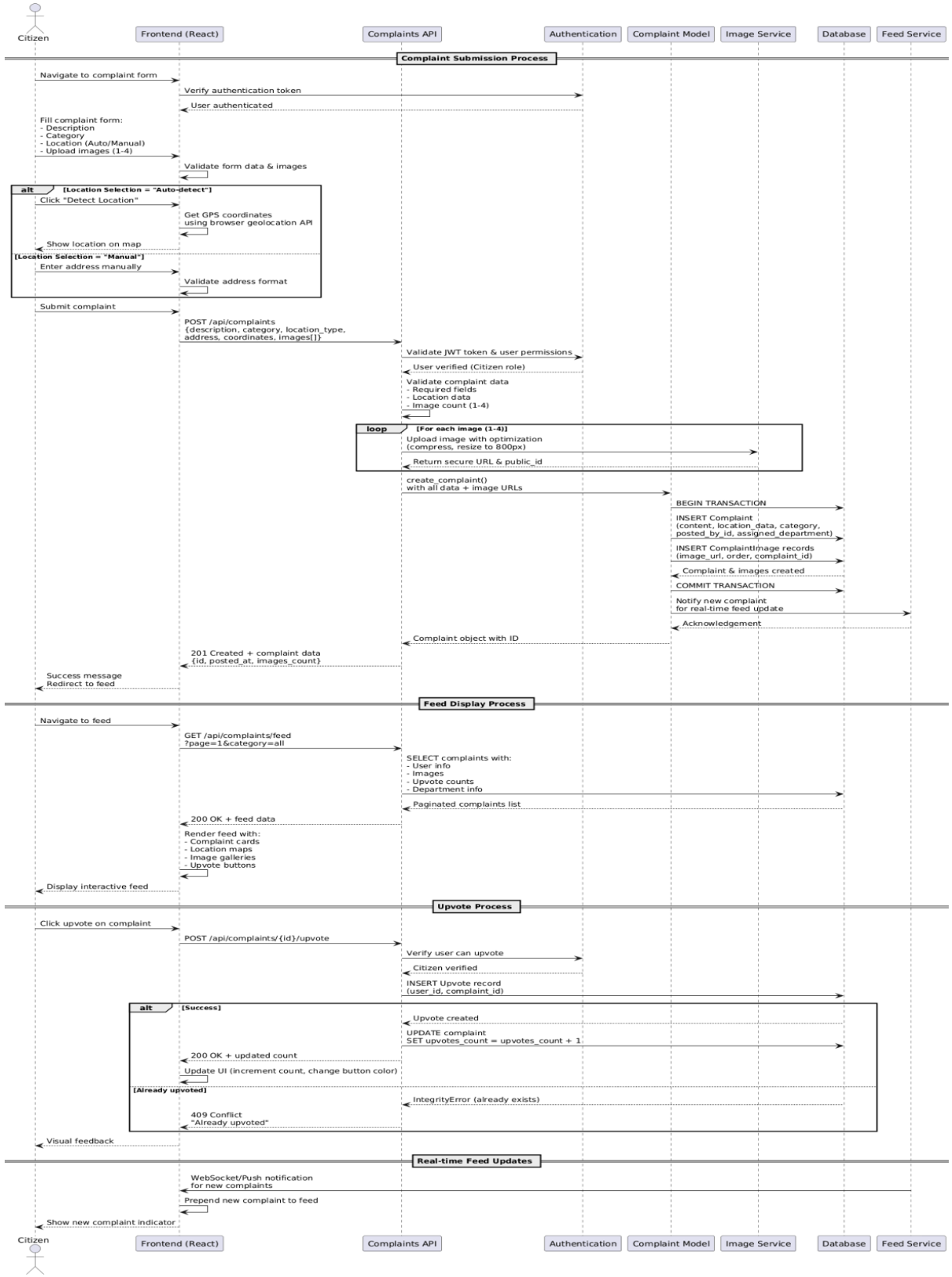
- Development of a **general complaint feed** accessible to both Citizens and Government Authorities.
- Enabling **upvotes** on complaints, allowing the community to indicate issues that impact them most.

This sprint enabled real-time civic reporting and laid the foundation for community-driven prioritization.

Class Diagram:



Sequence Diagram:



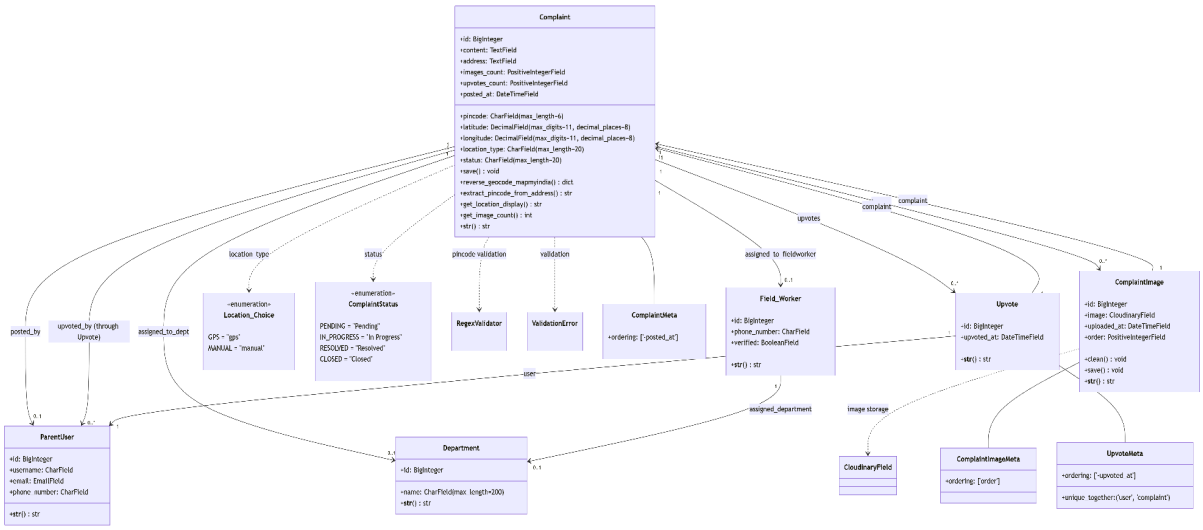
Sprint 3: Community Features & Past Complaints Workflow

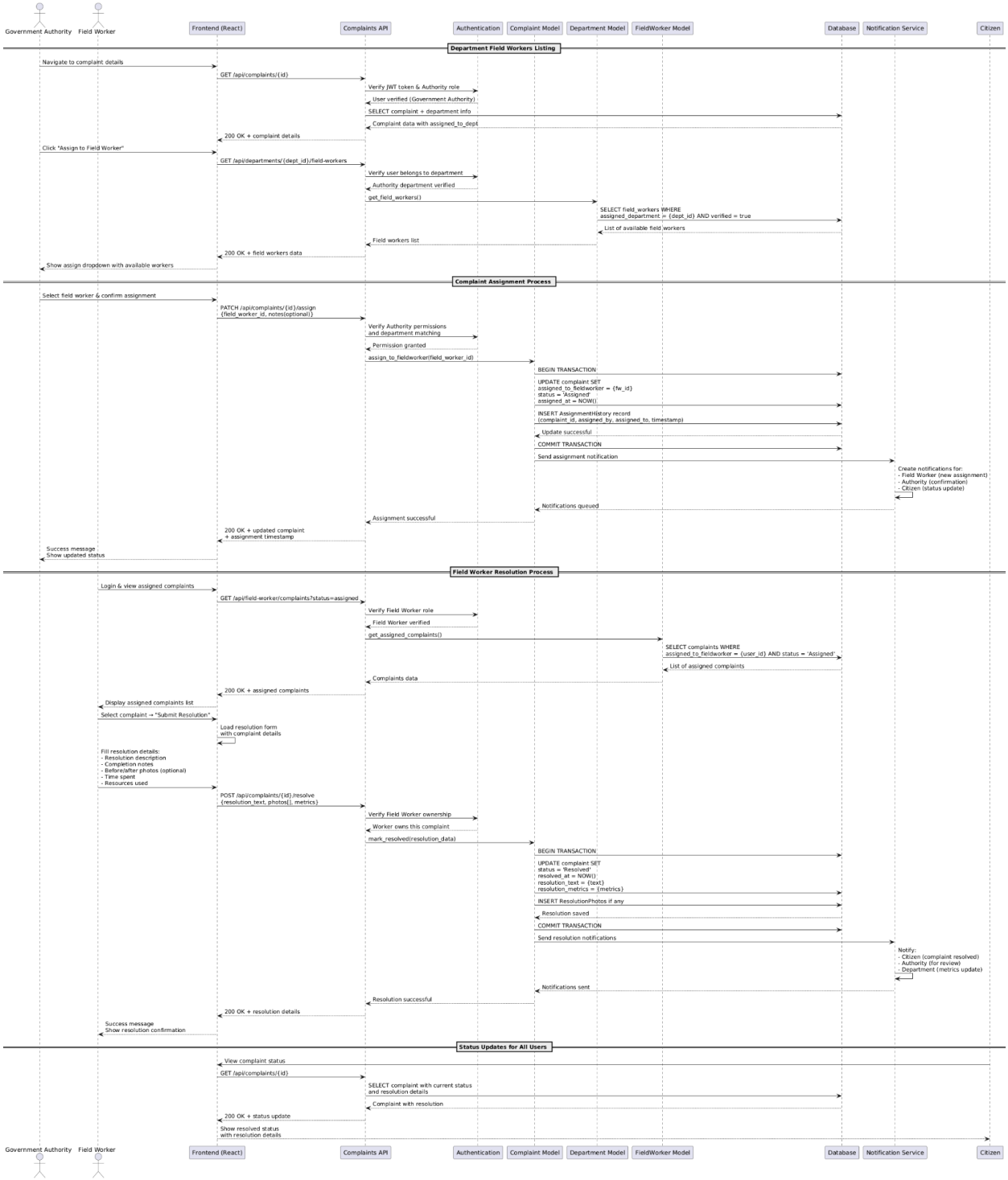
In Sprint 3, the team built features that connected the complaint submission system with the resolution workflow.

Key outcomes:

- Government Authorities were given the ability to **assign complaints** to appropriate **Field Workers** within their designated department.
- Field Workers received an interface to **view their assigned complaints** and **submit resolutions**.
- Existing functionalities underwent **testing**, ensuring data flow between Citizens, Authorities, and Workers functioned correctly.

This sprint established the operational cycle of reporting → assignment → resolution.




Government Authority
Field Worker
Frontend (React)
Complaints API
Authentication
Complaint Model
Department Model
FieldWorker Model
Database
Notification Service
Citizen
Department Field Workers Listing
Navigate to complaint details
GET /api/complaints/{id}
Verify JWT token & Authority role
User verified (Government Authority)
SELECT complaint + department info
Complaint data with assigned_to_dept
200 OK + complaint details
Click "Assign to Field Worker"
GET /api/departments/{dept_id}/field-workers
Verify user belongs to department
Authority department verified
get_field_workers()
SELECT field_workers WHERE
assigned_department = {dept_id} AND verified = true
List of available field workers
Field workers list
200 OK + field workers data
Show assign dropdown with available workers
Complaint Assignment Process
Select field worker & confirm assignment
PATCH /api/complaints/{id}/assign
{field_worker_id, notes(optional)}
Verify Authority permissions
and department matching
Permission granted
assign_to_fieldworker(field_worker_id)
BEGIN TRANSACTION
UPDATE complaint SET
assigned_to_fieldworker = {fw_id}
status = 'Assigned'
assigned_at = NOW()
INSERT AssignmentHistory record
(complaint_id, assigned_by, assigned_to, timestamp)
Update successful
COMMIT TRANSACTION
Send assignment notification
Create notifications for:
- Field Worker (new assignment)
- Authority (confirmation)
- Citizen (status update)
Notifications queued
Assignment successful
200 OK + updated complaint
+ assignment timestamp
Success message
Show updated status
Field Worker Resolution Process
Login & view assigned complaints
GET /api/field-worker/complaints?status=assigned
Verify Field Worker role
Field Worker verified
get_assigned_complaints()
SELECT complaints WHERE
assigned_to_fieldworker = {user_id} AND status = 'Assigned'
List of assigned complaints
Complaints data
200 OK + assigned complaints
Display assigned complaints list
Select complaint → "Submit Resolution"
Load resolution form
with complaint details
Fill resolution details:
- Resolution description
- Completion notes
- Before/after photos (optional)
- Time spent
- Resources used
POST /api/complaints/{id}/resolve
{resolution_text, photos[], metrics}
Verify Field Worker ownership
Worker owns this complaint
mark_resolved(resolution_data)
BEGIN TRANSACTION
UPDATE complaint SET
status = 'Resolved'
resolved_at = NOW()
resolution_text = {text}
resolution_metrics = {metrics}
INSERT ResolutionPhotos if any
Resolution saved
COMMIT TRANSACTION
Send resolution notifications
Notify:
- Citizen (complaint resolved)
- Authority (for review)
- Department (metrics update)
Notifications sent
Resolution successful
200 OK + resolution details
Success message
Show resolution confirmation
Status Updates for All Users
View complaint status
GET /api/complaints/{id}
SELECT complaint with current status
and resolution details
Complaint with resolution
200 OK + status update
Show resolved status
with resolution details

Sprint 4: Field Worker Resolution Cycle & Notifications

Sprint 4 expanded the system to support a more realistic workflow for resolution verification.

Main achievements:

- Citizens were allowed to **approve Field Worker resolutions**.
- If a Citizen was not satisfied, the complaint was **reset to its initial state**, enabling Government Authorities to **reassign** it to another Field Worker.
- A **notification system** was implemented to inform relevant users (Citizen, Field Worker, or Authority) about updates or tasks requiring attention.
- Comprehensive **unit testing** was performed for the newly added components.

This sprint added a feedback loop, ensuring resolution quality and continual engagement.

Sprint 5: AI-Based Department Suggestion & Time Estimation

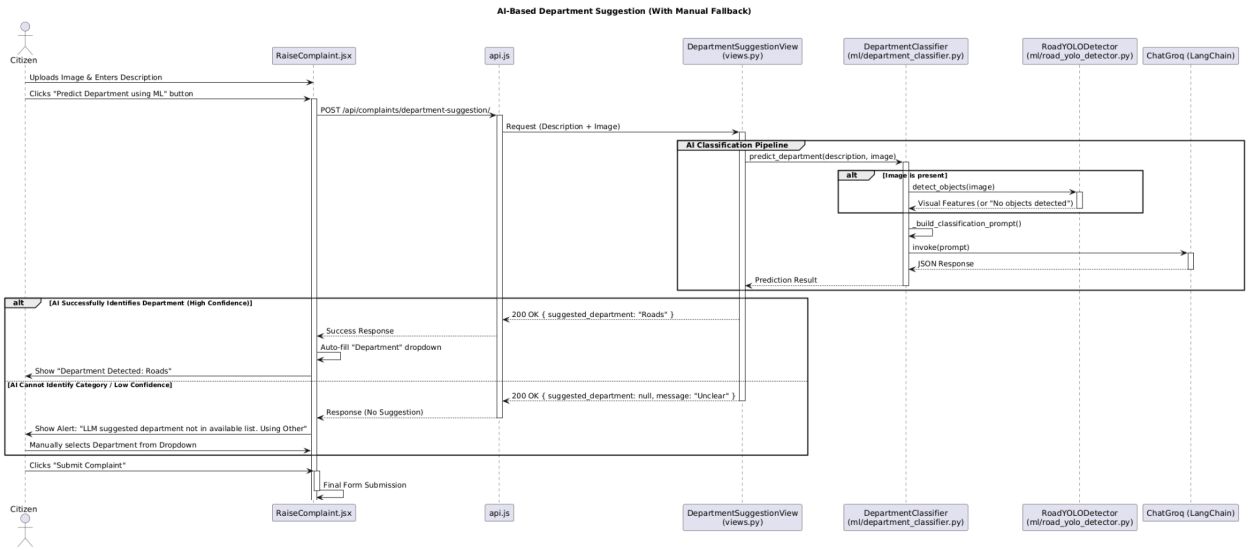
Sprint 5 focused on integrating AI to enhance decision-making and automate repetitive tasks.

Enhancements delivered:

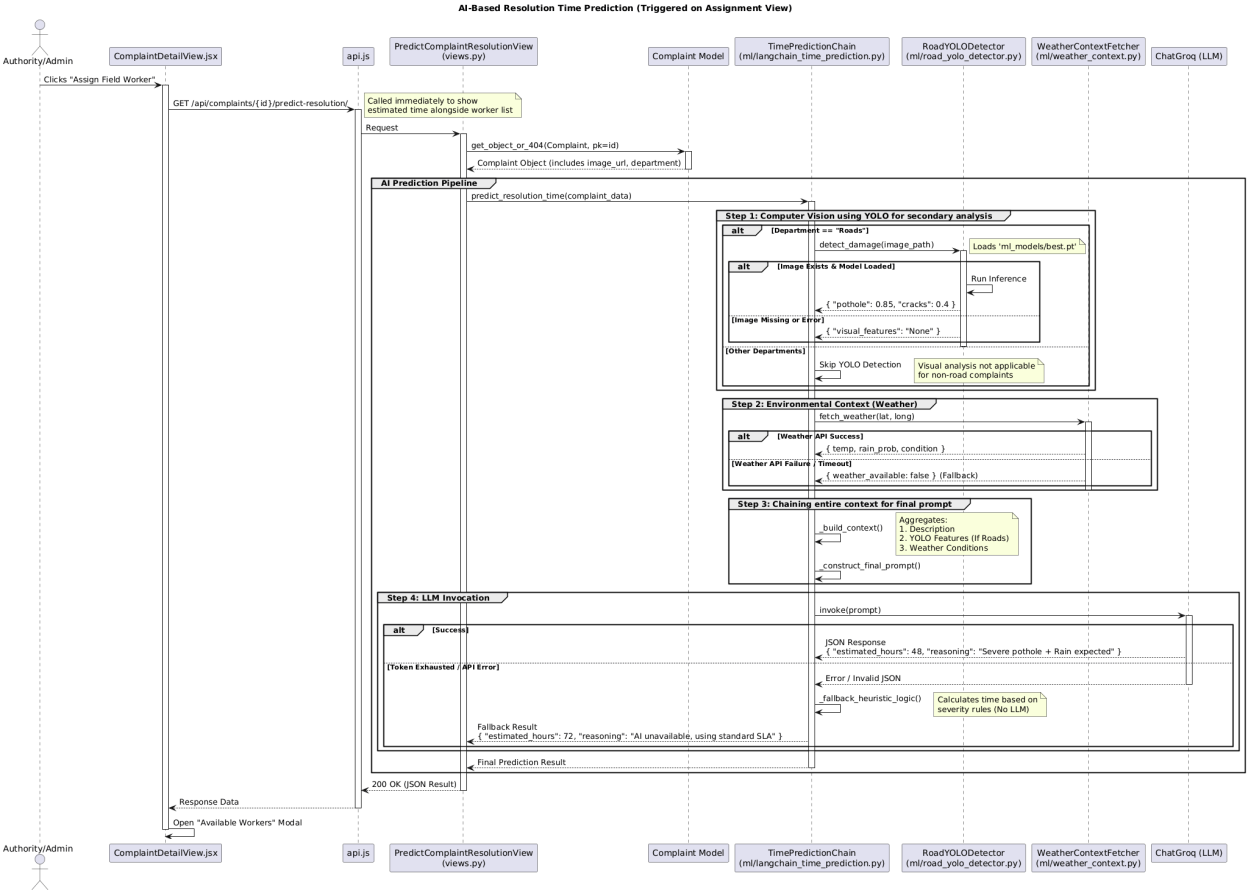
- When a user raised a complaint with an image, the AI system analyzed the visual content to **suggest the most appropriate department** to handle the issue.
- When the Government Authority assigned the complaint to a Field Worker, the AI model generated an **estimated time of completion**, along with a **brief explanatory justification**.

This sprint introduced intelligence into the system, making the platform smarter and more efficient.

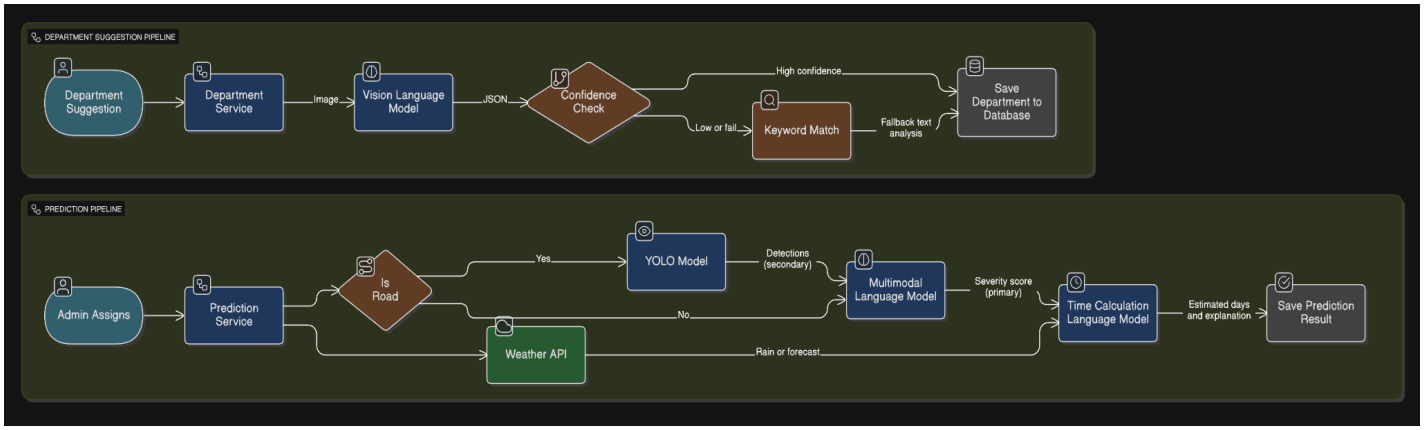
Department Suggestion Sequence Diagram:



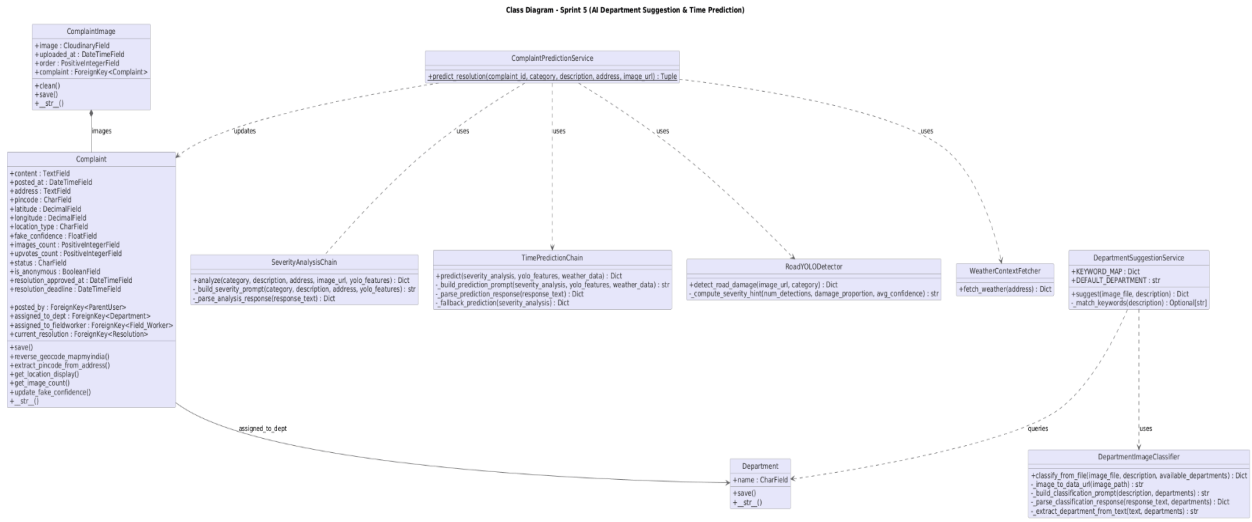
Time prediction pipeline Sequence Diagram:



High Level Flowchart:



Class Diagram:



Sprint 6: Verification, Testing, and Production Deployment Enhancements

The final sprint focused on validation, polishing, and preparing the platform for real-world deployment.

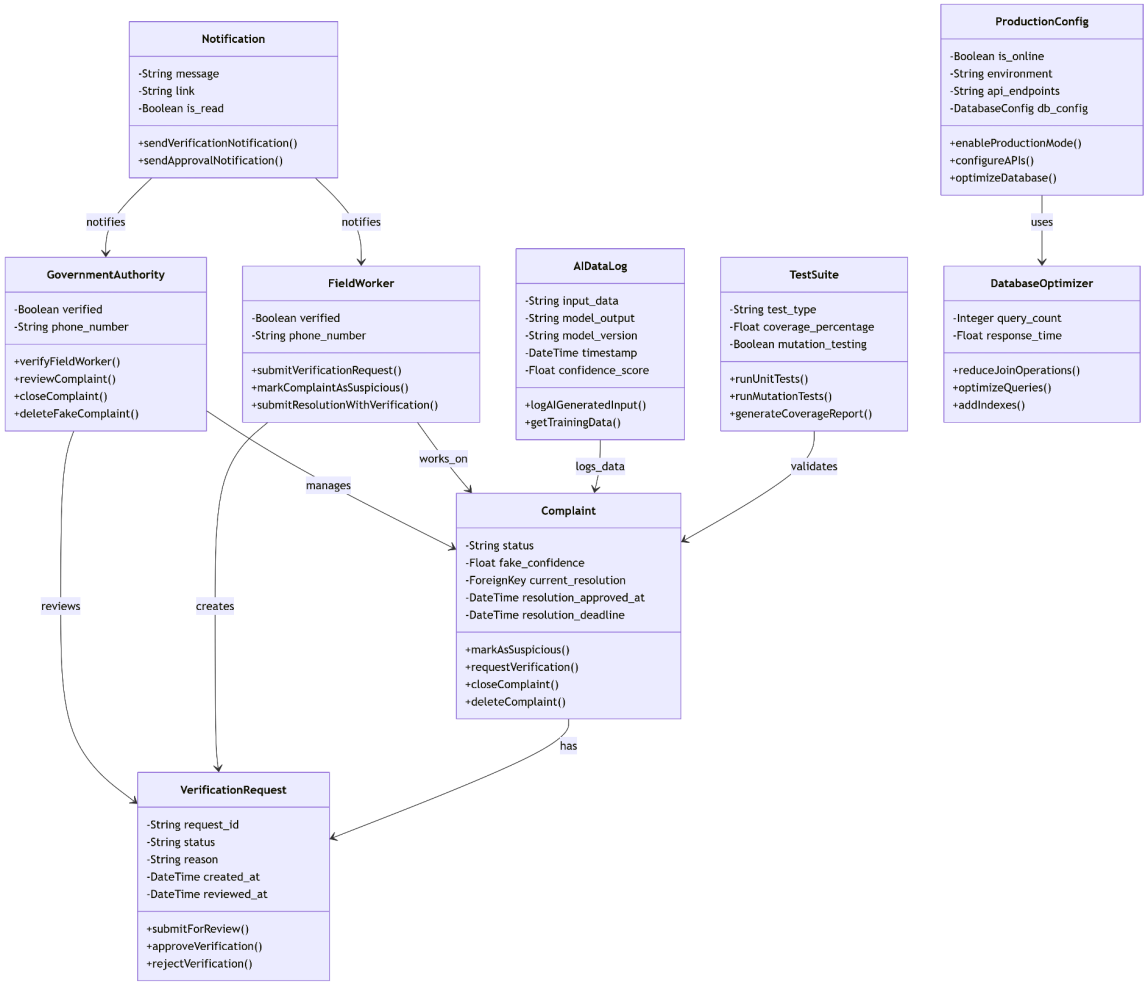
Completed tasks:

- Government Authorities were empowered to **verify field worker reports**, ensuring the issue was genuine before closing or deleting a complaint marked as fake.
- AI-generated inputs were **stored in the database** for future model refinement and analysis.

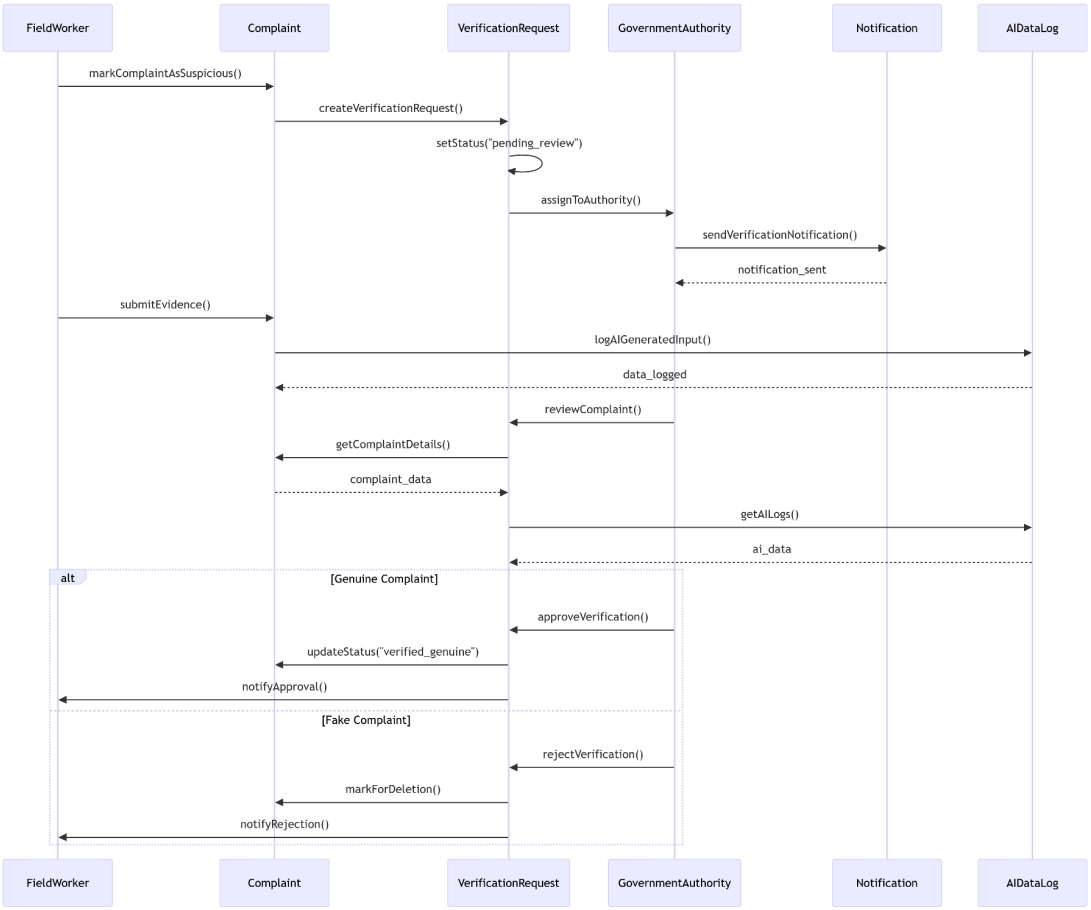
- Reduced database join calls to improve fetching complexity.
- Extensive **unit testing** and **mutation testing** were conducted, aiming for **100% coverage** to ensure system reliability.
- Backend and frontend were upgraded from **local (offline)** configurations to **production-ready (online)** mode.
- A workflow was enabled wherein Field Workers could **request verification** for complaints suspected to be fake, allowing Authorities to delete them after review.

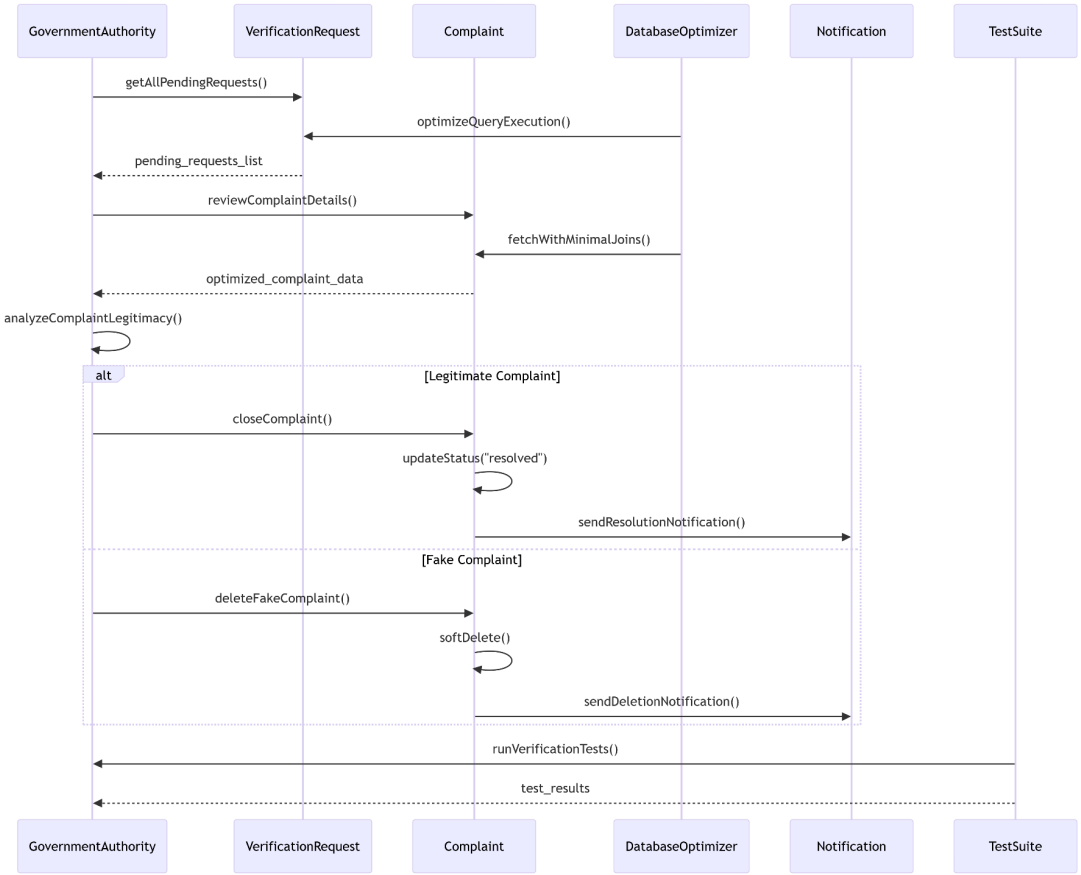
This sprint ensured the system met real-world performance, quality, and reliability expectations.

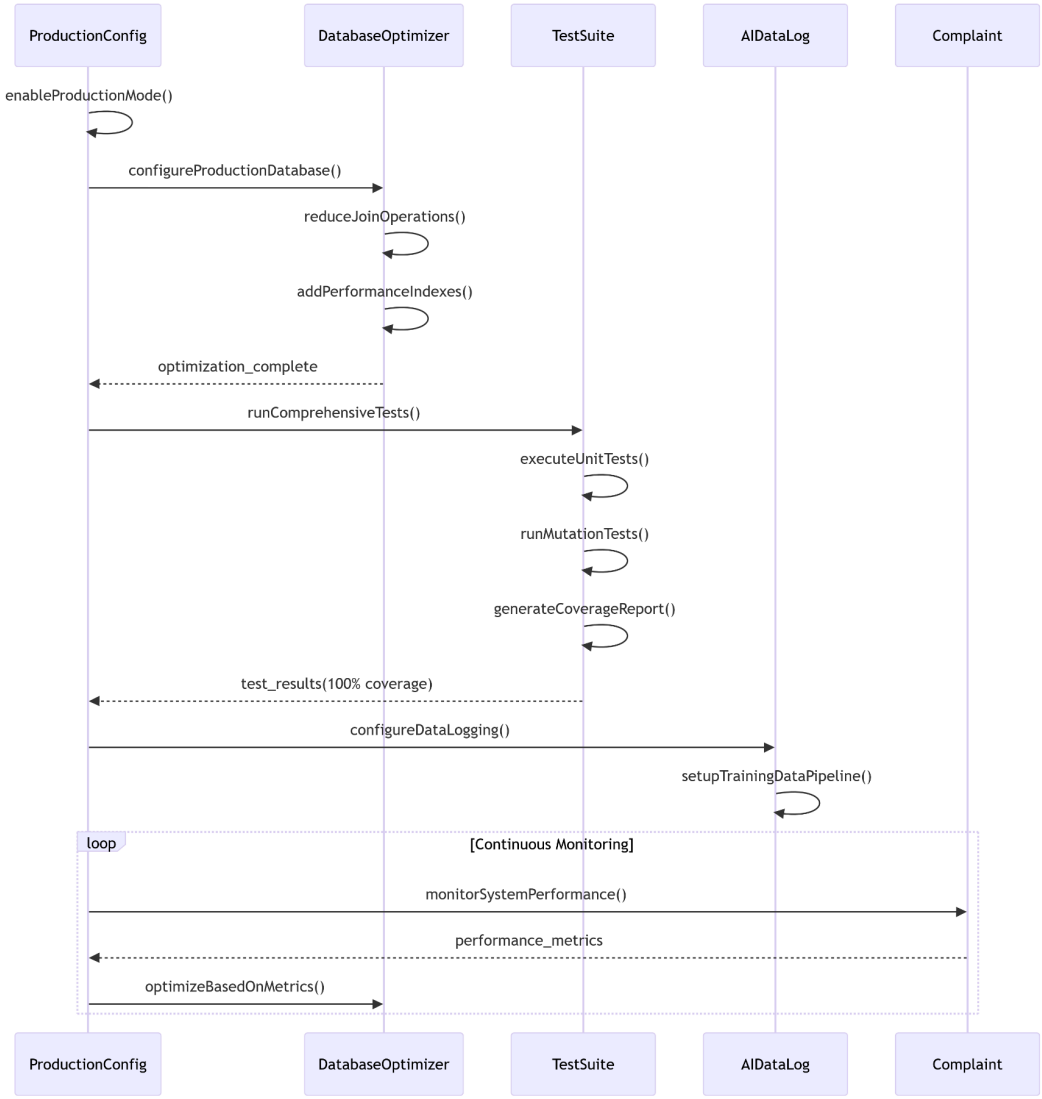
Class Diagram:

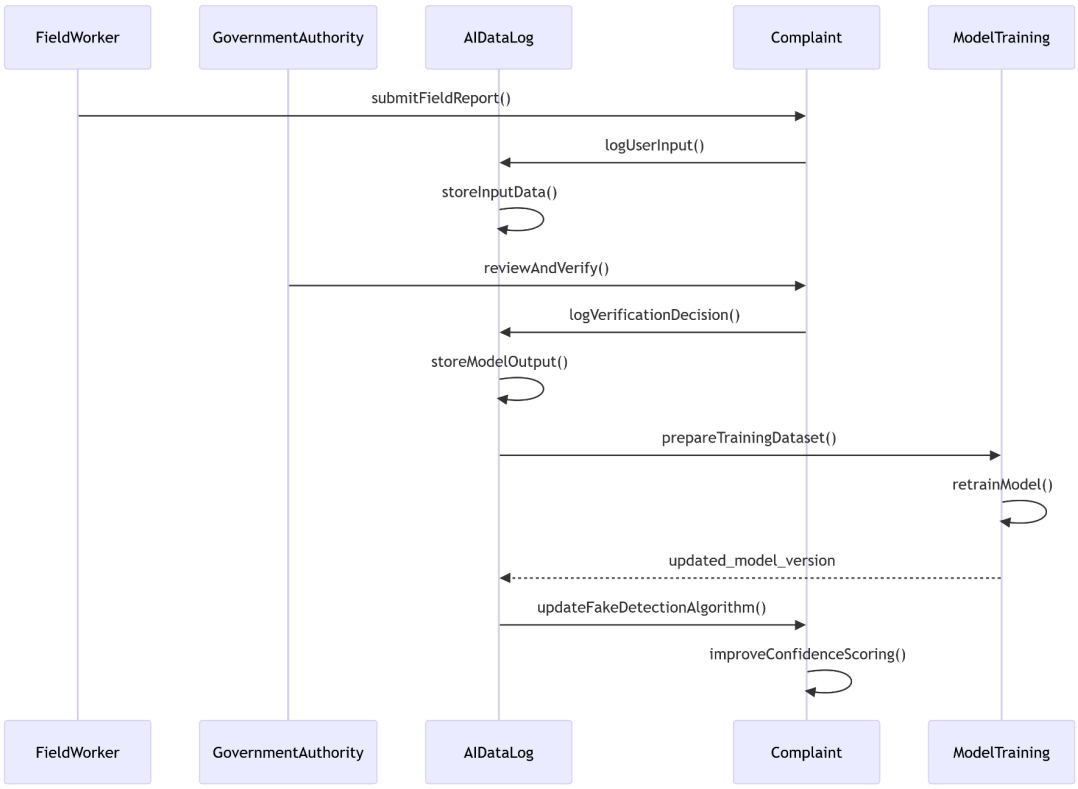


Sequence Diagram:




GovernmentAuthority
VerificationRequest
Complaint
DatabaseOptimizer
Notification
TestSuite
getAllPendingRequests()
optimizeQueryExecution()
pending_requests_list
reviewComplaintDetails()
fetchWithMinimalJoins()
optimized_complaint_data
analyzeComplaintLegitimacy()
alt
[Legitimate Complaint]
closeComplaint()
updateStatus("resolved")
sendResolutionNotification()
[Fake Complaint]
deleteFakeComplaint()
softDelete()
sendDeletionNotification()
runVerificationTests()
test_results
GovernmentAuthority
VerificationRequest
Complaint
DatabaseOptimizer
Notification
TestSuite


ProductionConfig
DatabaseOptimizer
TestSuite
AIDataLog
Complaint
enableProductionMode()
configureProductionDatabase()
reduceJoinOperations()
addPerformanceIndexes()
optimization_complete
runComprehensiveTests()
executeUnitTests()
runMutationTests()
generateCoverageReport()
test_results(100% coverage)
configureDataLogging()
setupTrainingDataPipeline()
loop
[Continuous Monitoring]
monitorSystemPerformance()
performance_metrics
optimizeBasedOnMetrics()
ProductionConfig
DatabaseOptimizer
TestSuite
AIDataLog
Complaint


FieldWorker
GovernmentAuthority
AIDataLog
Complaint
ModelTraining
submitFieldReport()
logUserInput()
storeInputData()
reviewAndVerify()
logVerificationDecision()
storeModelOutput()
prepareTrainingDataset()
retrainModel()
updated_model_version
updateFakeDetectionAlgorithm()
improveConfidenceScoring()
FieldWorker
GovernmentAuthority
AIDataLog
Complaint
ModelTraining