FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

System Modelling and Design for a Mobile Network Monitoring Application in Cameroon

COURSE MASTER: Dr. NKEMENI VALERY

COURSE TITLE: Internet Programming and Mobile Programming

COURSE CODE: CEF 440

GROUP 23

S/N	NAME	MATRICULE
1	KINENUI STEPH	FE22A244
2	TIAYA FOTSEU JOSUE	FE22A315
3	NGANYU BRANDON YUNIWO	FE22A258
4	NGULEFAC FOLEFAC FOBELLA	FE19081
5	JAMISON LII	FE22A284

Table of Contents

Introduction

System modelling and design constitute fundamental phases in the software engineering process, essential for developing clear, well-structured, and reliable software systems. These phases facilitate a comprehensive understanding of system requirements, define the interactions between components, and establish a blueprint for subsequent development and implementation. By employing formal models and diagrammatic representations, stakeholders—including developers, designers, and clients—are able to communicate effectively and detect potential issues early, ultimately reducing costs and improving software quality. Furthermore, system modelling ensures alignment with user needs and technical constraints, fostering a systematic approach to software construction and maintenance.

This report focuses on the system modelling and design of a mobile network monitoring application that leverages intelligent Quality of Experience (QoE) metrics to enhance user satisfaction. The application serves as a sophisticated tool enabling mobile network users to perform multiple critical functions: testing network speed, reporting connectivity issues, reviewing historical usage data, configuring privacy settings, and engaging in communication with system administrators and mobile network providers. These features collectively empower users to gain transparent insights into network performance and actively participate in ensuring optimal service delivery.

The application's architecture involves multiple key actors, each playing a distinct role within the system workflow:

- **Mobile Network User:** The primary end-user who interacts with the application to monitor network conditions, submit issue reports, and access personal usage history.
- **System Administrator:** Responsible for managing and maintaining the application's infrastructure, overseeing user data security, and facilitating communication channels between users and providers.
- **Mobile Network Provider:** The service entity that receives user feedback and performance metrics to optimize network coverage and quality of service.

Through systematic modelling and design, this report aims to elucidate the structural and behavioral aspects of the mobile network monitoring system. The use of standardized diagrams and formal modelling techniques will provide clear documentation and support efficient implementation, ensuring that the system meets both functional and non-functional requirements.

Context Diagram

A *context diagram* is a high-level, visual representation that defines the scope and boundaries of a system by illustrating the system as a single process and showing its interactions with external entities. It acts as an essential tool in system modelling by providing a simplified overview of how the system interfaces with users, other systems, and external data sources. This diagram aids stakeholders in understanding system boundaries and data exchanges without delving into internal details.

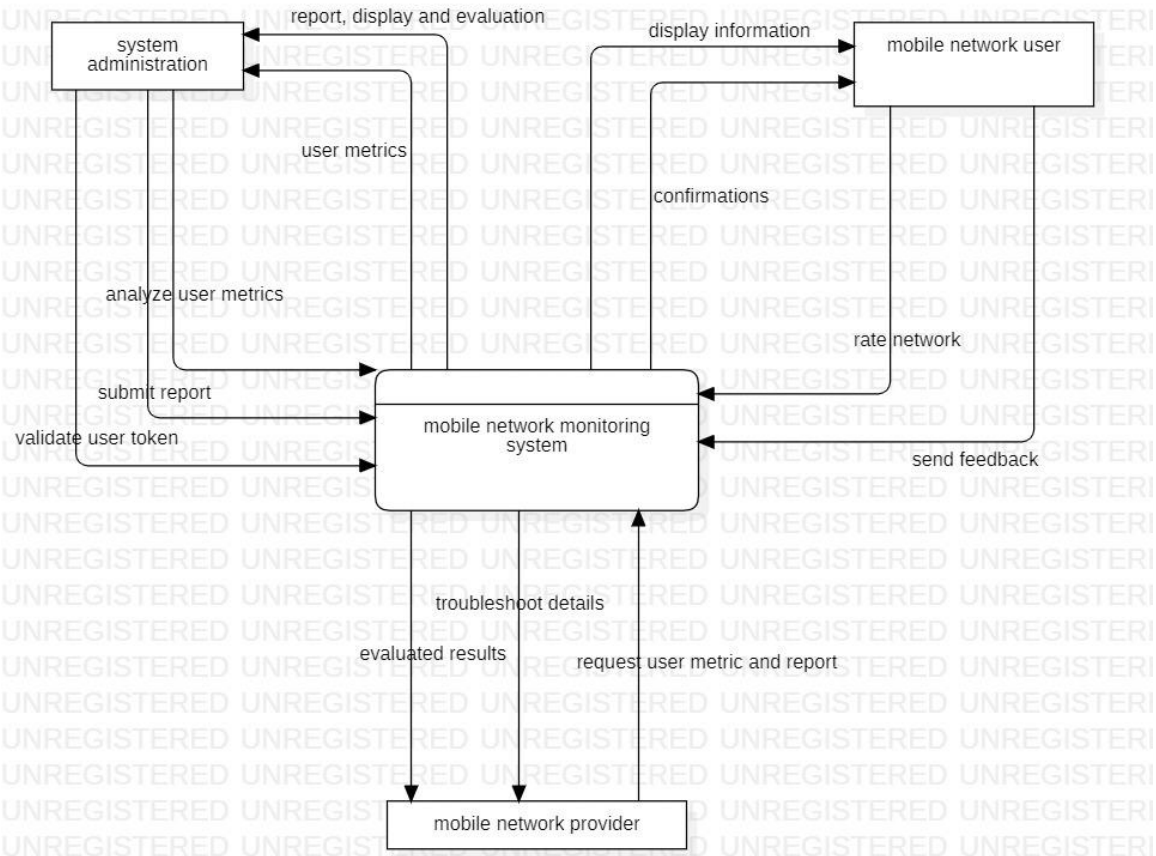
For the mobile network monitoring application, the context diagram identifies three primary external entities:

- **Mobile Network User:** The individual using the application to test network speed, report issues, review usage history, and configure preferences.
- **System Administrator:** The internal actor responsible for system management, user support, and maintenance tasks.
- **Mobile Network Provider:** The external service provider who receives aggregated feedback and network performance data to enhance service quality.

The major data flows between these entities and the system define essential communication pathways:

- **From Mobile Network User to System:** Inputs such as speed test requests, issue reports, and user preferences.
- **From System to Mobile Network User:** Outputs including test results, notifications, historical data views, and responses to issue reports.
- **Between System and System Administrator:** Flow of maintenance commands, configuration data, and user management information.
- **From System to Mobile Network Provider:** Aggregated network performance metrics and summarized issue reports for network optimization.
- **From Mobile Network Provider to System:** Updates related to network status changes, maintenance schedules, or configuration parameters.

These entities and data flows were selected deliberately to encapsulate the main interfaces that the system must support, ensuring clarity in system boundaries. By clearly defining the scope, the context diagram prevents ambiguity, facilitates requirements validation, and establishes a foundation for more detailed modelling stages. It also helps in aligning stakeholder expectations by demonstrating what lies inside and outside the system's control.



Dataflow Diagram (DFD)

A *Dataflow Diagram (DFD)* is a graphical representation used to illustrate the flow of data within a system, showcasing how input data is transformed into output results through various processes. It provides a clear visualization of how information moves between system components, external entities, and data stores, enabling an in-depth understanding of system functionality and interactions. DFDs are instrumental in identifying bottlenecks, redundancies, and inefficiencies in data handling, thereby supporting effective system analysis and design.

In this project, two levels of DFD are employed to progressively detail the system's operations:

- **Context-Level DFD:** This highest level gives an overview of the entire mobile network monitoring application as a single process, highlighting its interactions with external entities such as the Mobile Network User, System Administrator, and Mobile Network Provider. It establishes the main data exchange pathways without delving into internal processing.
- **Level 1 DFD (Decomposition):** This level breaks down the main system process into multiple distinct subprocesses, each responsible for specific functionality such as conducting speed tests, managing issue reports, and handling user

authentication. It also introduces internal data stores for persisting user data and network metrics.

Key processes identified in the Level 1 DFD include:

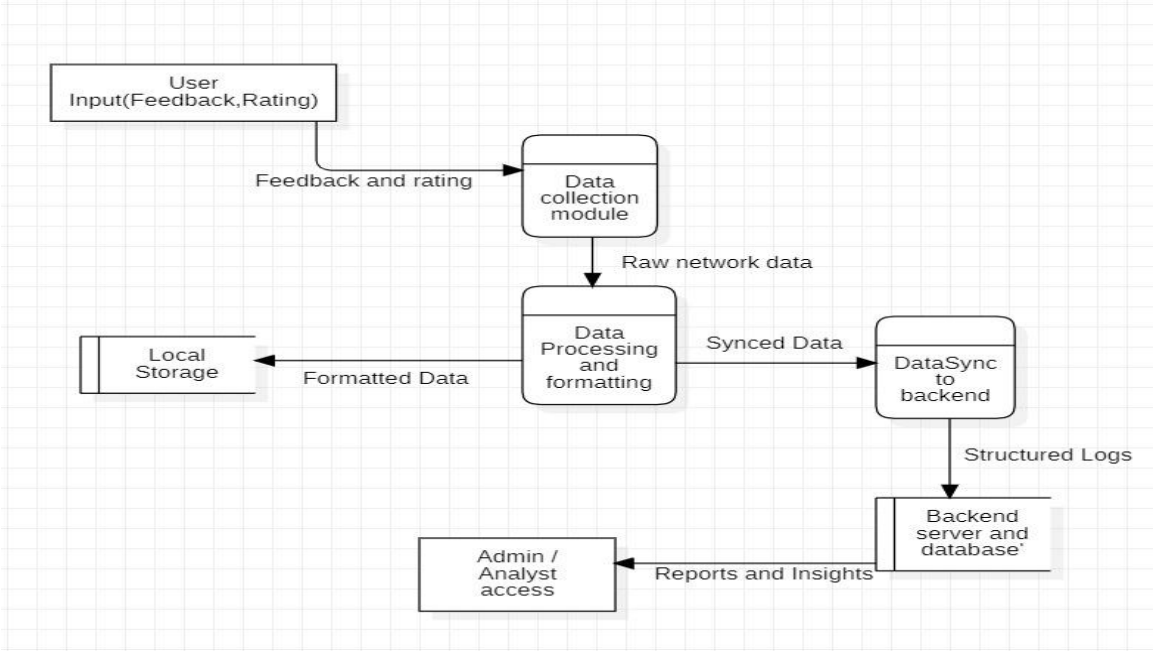
- **Speed Test Processing:** Manages user requests to perform network speed tests, interfaces with measurement tools, and returns tested results to users.
- **Issue Reporting:** Collects user-submitted connectivity problems, validates input, stores reports, and forwards critical issues to administrators and providers for resolution.
- **User Account Management:** Handles user login, privacy settings configuration, and access control to ensure secure and personalized experiences.
- **Data Analysis and Reporting:** Aggregates network performance data over time, providing historical views and QoE metrics that assist both users and providers.

The DFD also incorporates *data stores* essential for system persistence:

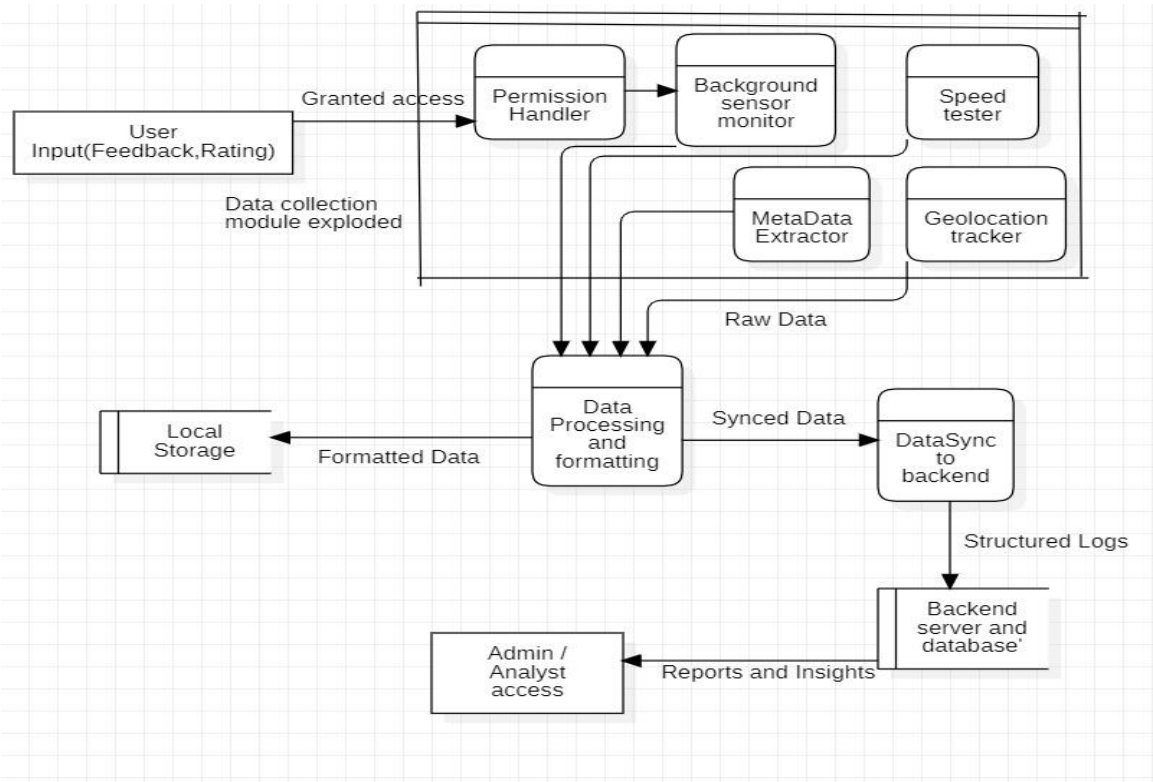
- **User Data Store:** Contains user profiles, preferences, and authentication details.
- **Network Performance Store:** Archives speed test results and aggregated network metrics.
- **Issue Reports Store:** Records submitted connectivity problems and their resolution status.

Data flows explicitly demonstrate how information is exchanged between processes, data stores, and external actors. For example, the mobile network user initiates a speed test request, which flows into the Speed Test Processing subprocess; the test results flow back to the user and are simultaneously recorded in the Network Performance Store. Similarly, issue reports move from users through the Issue Reporting process to both the Issue Reports Store and onward to administrators for action.

The design of these processes and data flows is intentional to ensure modularity, security, and efficiency. By delineating responsibilities, the system can handle diverse functionalities concurrently while maintaining data integrity and responsiveness. The DFD aids stakeholders by offering a structured view of system operations, enhancing communication between developers, analysts, and users, and facilitating validation of requirements against actual data movement and processing.



Level 1 Data flow



Level 2 Data flow

Use Case Diagram

The *Use Case Diagram* plays a pivotal role in system modelling by capturing the system's functional requirements from the perspective of its users. It graphically represents the interactions between users (actors) and the system, thereby providing a clear, high-level overview of the services the system must deliver. This user-centric focus supports the identification and organisation of system functionalities in a manner that is accessible to both technical and non-technical stakeholders, facilitating consensus and mutual understanding during the requirements elicitation and validation phases.

For the mobile network monitoring application, the use case diagram delineates several key **actors**:

- **Mobile Network User:** The principal end-user who engages with the application to monitor and manage personal network experiences.
- **System Administrator:** The internal actor responsible for operational oversight, user management, and system maintenance tasks.
- **Mobile Network Provider:** The external service entity that interfaces with the system for receiving issue reports and network performance data.

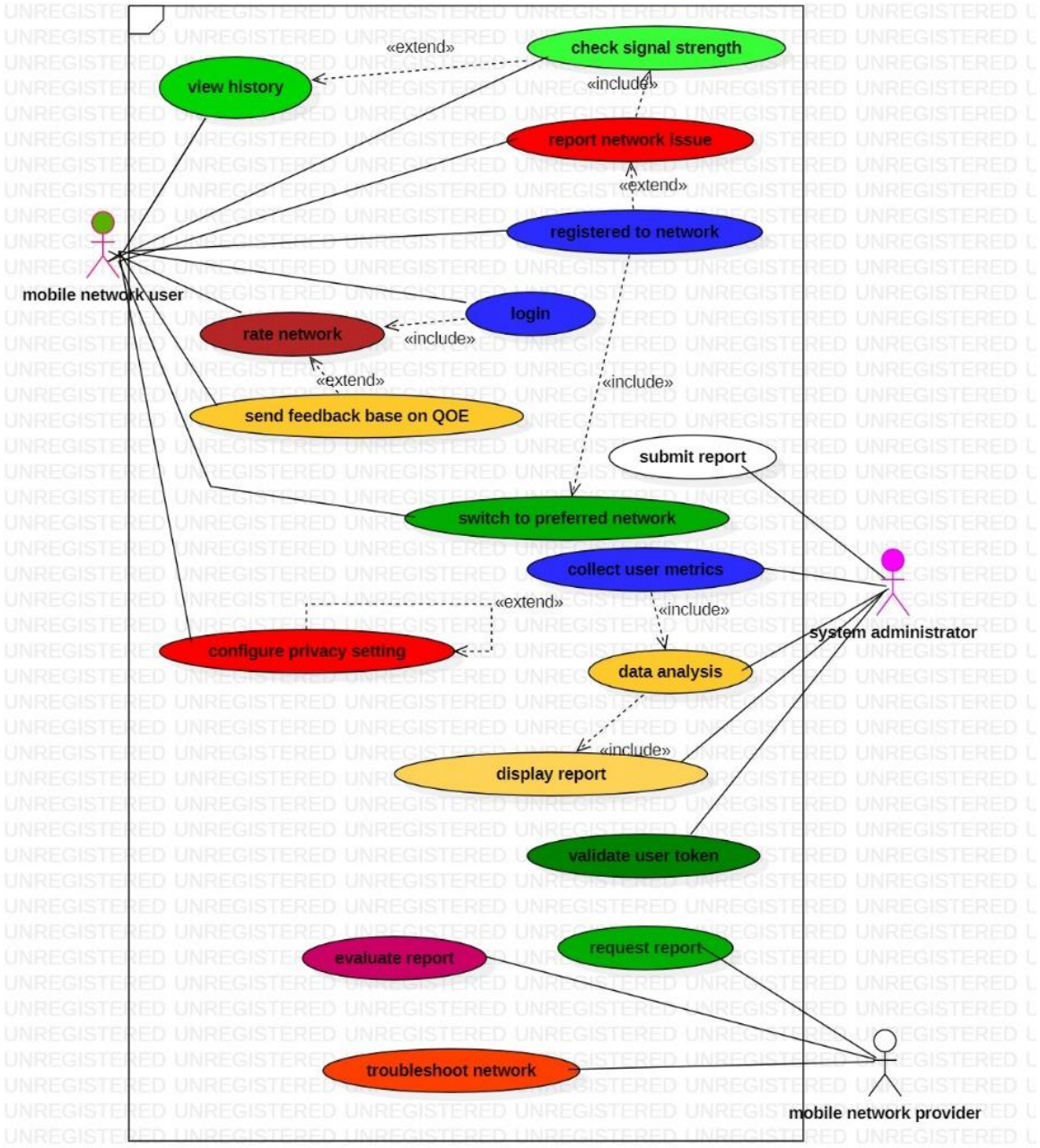
The principal **use cases** encapsulate critical functionalities that satisfy both user needs and system requirements:

- **Login:** Enables secure authentication for users and administrators, ensuring access control and personalized experiences.
- **Test Speed:** Allows users to initiate network speed tests, retrieving real-time performance metrics for analysis.
- **Report Issue:** Facilitates submission of connectivity problems or anomalies, which are logged and escalated to administrators and providers for resolution.
- **View History:** Provides users with access to their historical usage data and past speed test results, empowering informed decision-making.
- **Analyze Data:** Supports administrators and providers in examining aggregated network metrics to identify trends and optimize service quality.
- **Manage User Accounts:** Allows administrators to configure user privileges, privacy settings, and system parameters.
- **Communicate with Providers:** Enables structured feedback exchange between the system and mobile network providers, facilitating collaborative improvements.

These use cases collectively underpin the core functionality of the mobile network monitoring application, ensuring that user interactions and administrative operations are clearly articulated and structured. The use case diagram thereby serves as a foundational artefact in bridging user requirements with system design, promoting traceability and alignment throughout the software development lifecycle.

By formalising actor roles and use cases, this diagram not only clarifies expectations for developers and stakeholders but also aids in identifying essential system services,

potential extensions, and boundary conditions. Consequently, it enhances communication, reduces misunderstandings, and supports rigorous validation of functional requirements.



Sequence Diagram

A *sequence diagram* is a dynamic modelling tool used to illustrate how objects and components within a system interact over time to carry out specific functionalities. It captures the sequence of messages exchanged among system entities, detailing the order in which operations occur and the flow of information. Sequence diagrams are particularly valuable in visualising the real-time behaviour of the system's processes, clarifying temporal dependencies and communication pathways that underpin core use cases.

For the mobile network monitoring application, two primary scenarios are modelled through sequence diagrams: **reporting a network issue** and **performing a speed test**. These scenarios were chosen to demonstrate key interactive workflows involving multiple actors and system components, highlighting both user engagement and backend processing.

Reporting a Network Issue

In this scenario, the **Mobile Network User** initiates the process by submitting an issue report via the application interface. The sequence diagram captures the flow as follows:

- The user sends an *issue report request* to the system's Issue Reporting component.
- The component validates the input data, checking for completeness and consistency.
- Upon successful validation, the issue report is stored in the *Issue Reports Store*, ensuring persistence.
- The system generates a confirmation message and sends it back to the user, acknowledging receipt.
- Simultaneously, a notification is forwarded to the **System Administrator** and the **Mobile Network Provider** for investigation and resolution.

This interaction sequence ensures timely and reliable handling of connectivity problems, promoting responsive user support and facilitating proactive network maintenance. The design choice to involve both administrators and providers in the notification flow supports collaborative issue management.

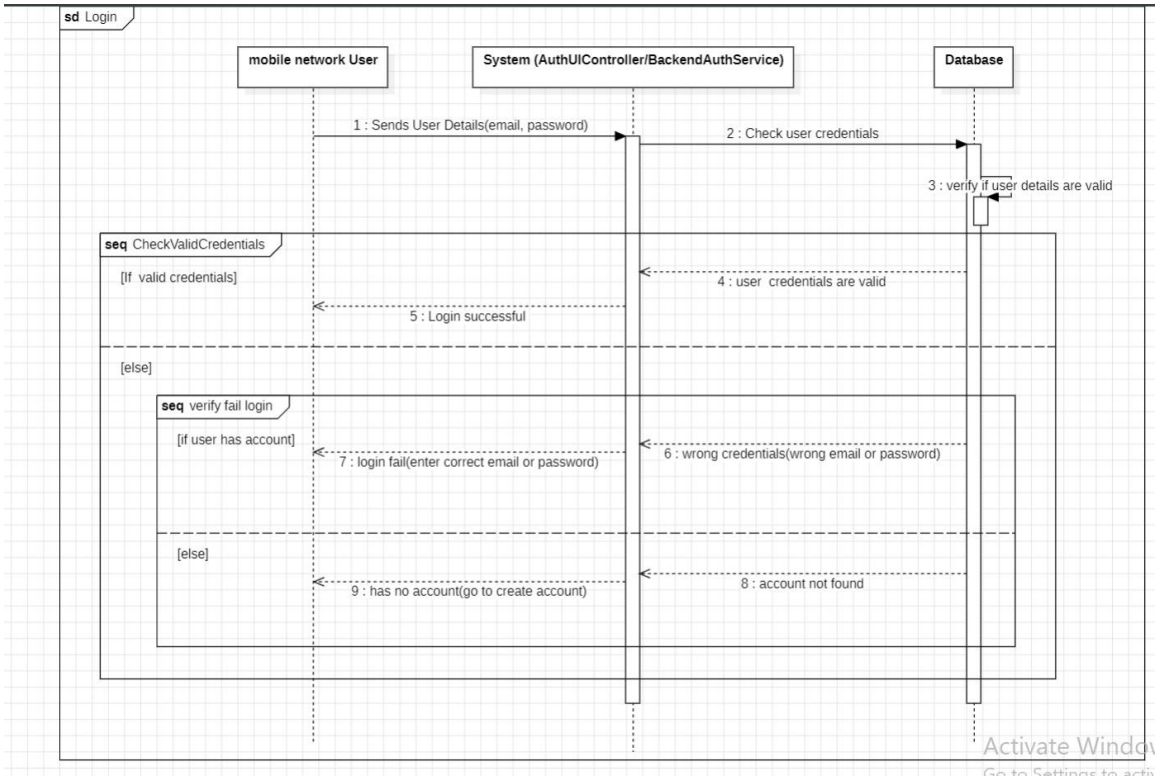
User Login and User Metrics Collection

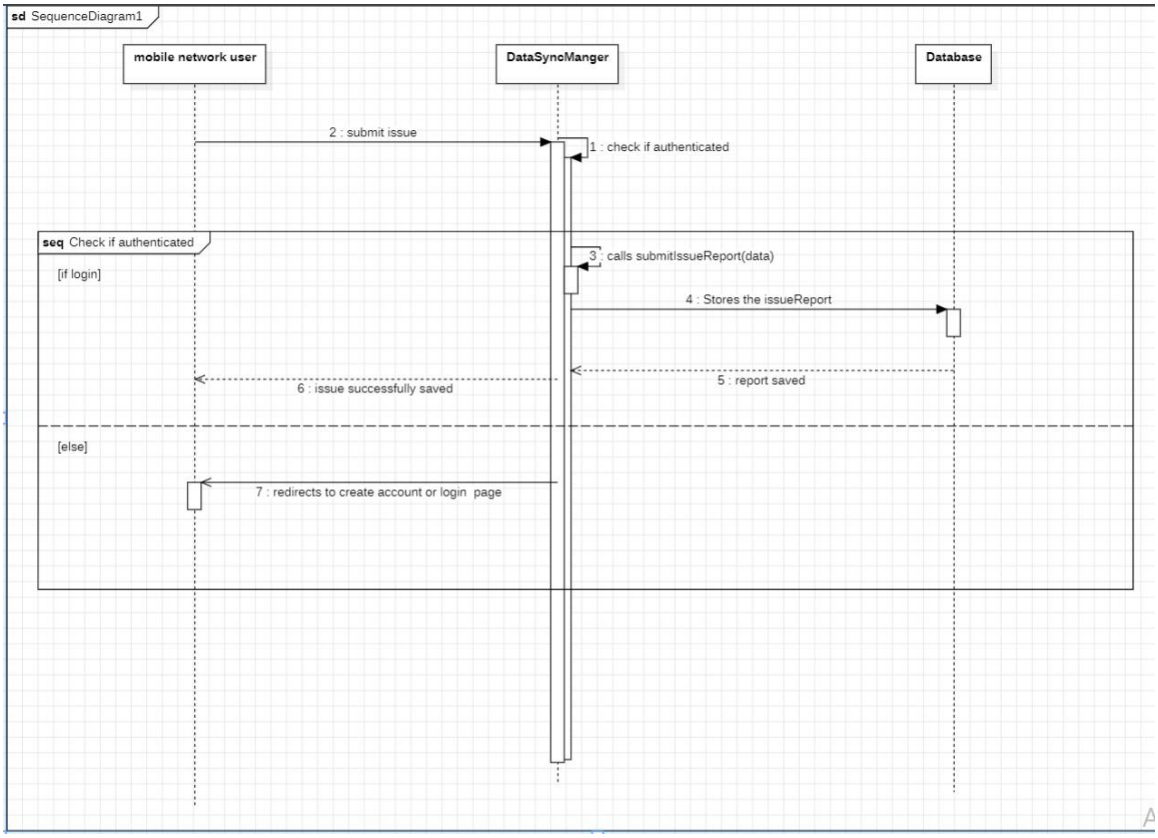
This scenario describes the interaction when a Mobile Network User accesses the application by logging in and then triggers the collection of user metrics. The flow proceeds as follows:

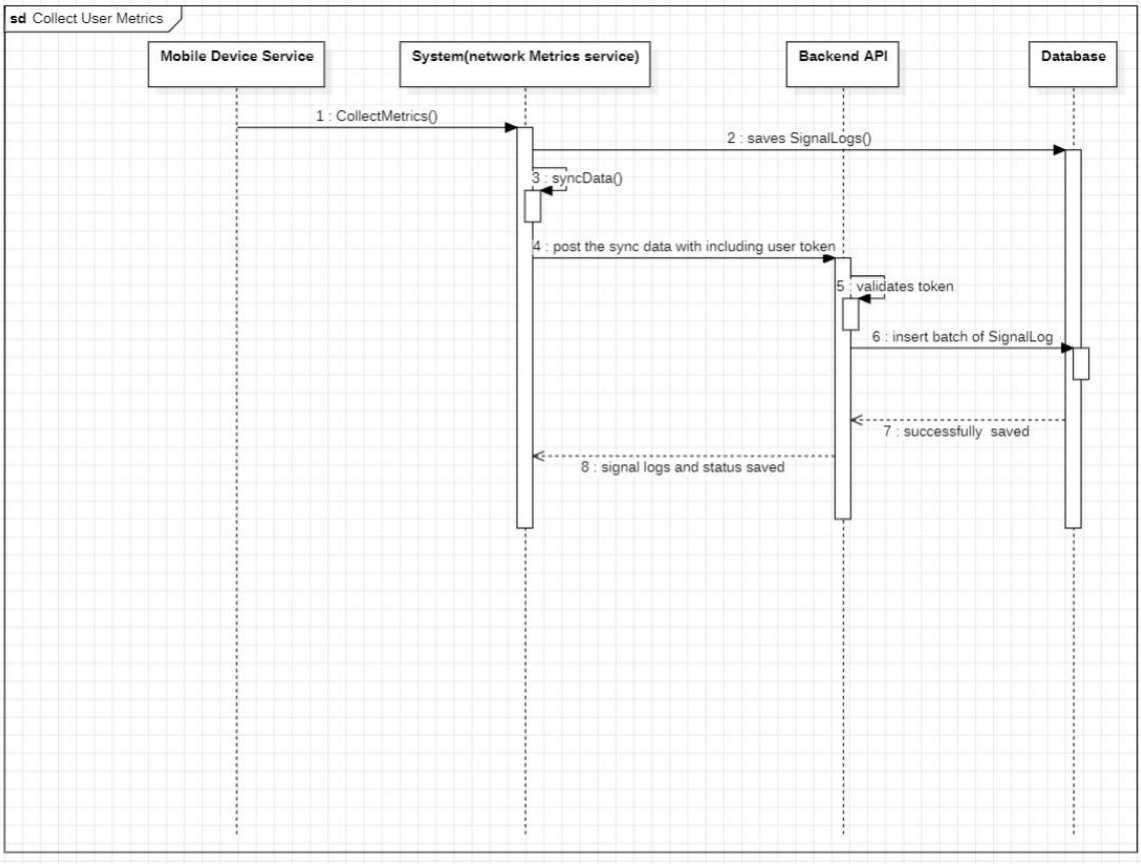
- The user sends a login request with credentials to the system's authentication component.
- The authentication component validates the credentials against the User Data Store.

- Upon successful authentication, a login confirmation is returned to the user, granting access.
- Immediately after login, the system initiates collection of user metrics, such as network status and usage statistics, by querying internal monitoring modules.
- The collected metrics are aggregated and stored in the Network Performance Store.
- The system optionally sends the metrics summary back to the user interface for real-time display or feedback purposes.

This sequence ensures secure access and provides the user with updated performance insights soon after authentication, enhancing the application's interactivity and responsiveness




sd SequenceDiagram1
mobile network user
DataSyncManger
Database
2 : submit issue
1 : check if authenticated
seq Check if authenticated
[if login]
3 : calls submitIssueReport(data)
4 : Stores the issueReport
6 : issue successfully saved
5 : report saved
[else]
7 : redirects to create account or login page


sd Collect User Metrics
Mobile Device Service
System(network Metrics service)
Backend API
Database
1 : CollectMetrics()
2 : saves SignalLogs()
3 : syncData()
4 : post the sync data with including user token
5 : validates token
6 : insert batch of SignalLog
7 : successfully saved
8 : signal logs and status saved

Class Diagram

The *class diagram* serves as a fundamental structural model that delineates the system's architecture through a detailed representation of classes, their attributes, methods, and the relationships among them. It provides an object-oriented perspective on the system design, capturing key entities and encapsulating both data and behaviour within distinct modules. This model facilitates a deeper understanding of the system components, promoting modularity, reusability, and maintainability.

In the mobile network monitoring application, several pivotal classes have been identified to represent critical system entities:

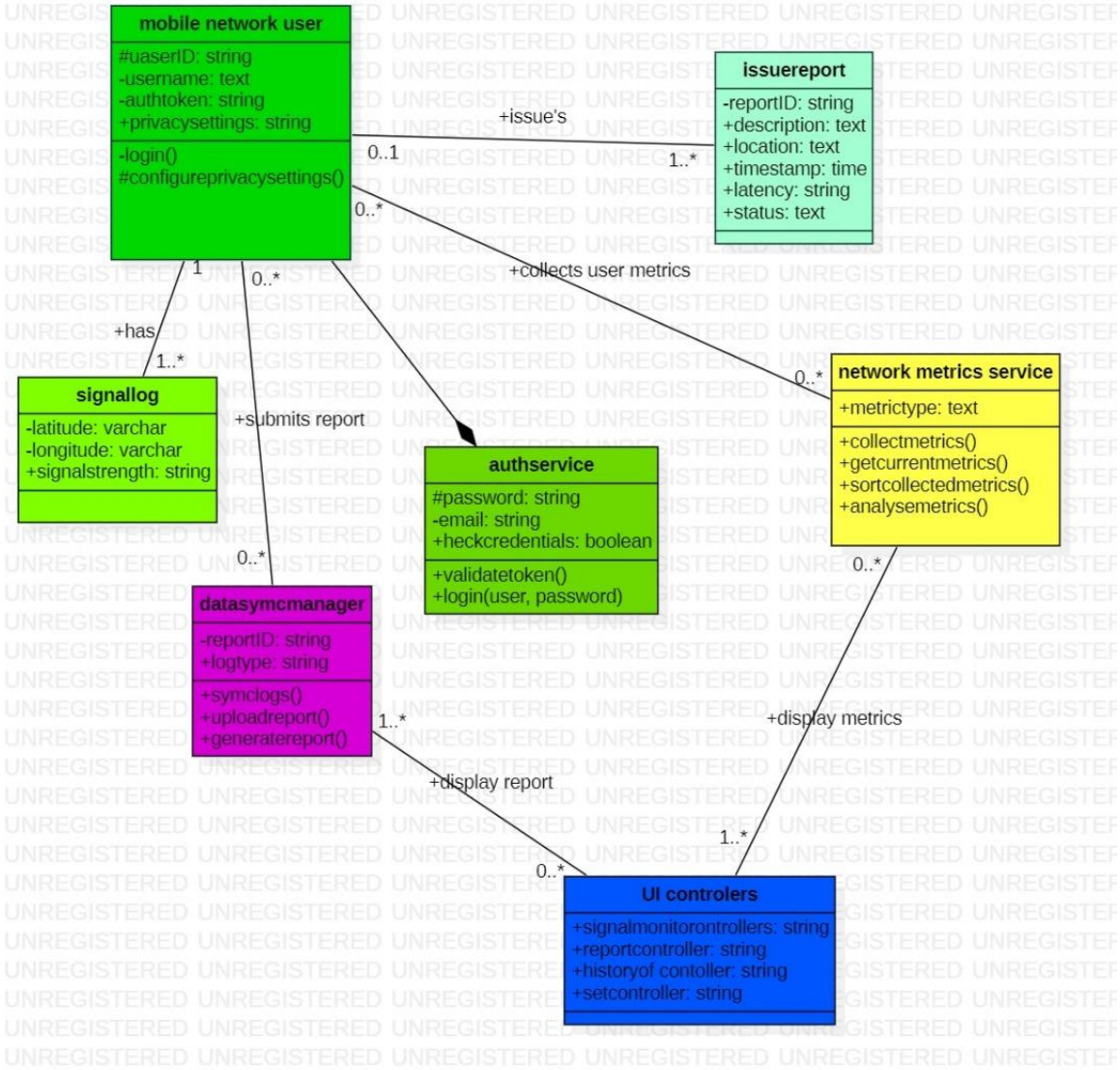
- **MobileNetworkUser:** This class encapsulates attributes such as *userID*, *username*, *contactInfo*, and *privacySettings*. Its methods include *authenticate()*, *requestSpeedTest()*, *submitIssueReport()*, and *viewUsageHistory()*, embodying the functional capabilities accessible to end-users.
- **SignalLog:** Responsible for recording network performance metrics, this class maintains attributes like *timestamp*, *downloadSpeed*, *uploadSpeed*, and *latency*. Methods such as *logSignalData()* and *retrieveMetrics()* handle data persistence and retrieval for analytical purposes.
- **IssueReport:** This class models connectivity problems submitted by users, with attributes including *reportID*, *userID*, *issueDescription*, *status*, and *resolutionComments*. Core methods are *createReport()*, *updateStatus()*, and *notifyAdministrator()*, supporting the lifecycle of issue management.
- **NetworkMetricsService:** Acting as a service layer, it aggregates and analyses SignalLog data. Its attributes include references to collections of signal logs and issue reports. Methods like *calculateQoE()*, *generatePerformanceSummary()*, and *sendDataToProvider()* enable synthesis of performance insights and communication with external entities.
- **AuthService:** This class manages authentication and authorization, containing attributes such as *authToken* and *sessionTimeout*. Its methods *login()*, *logout()*, and *validateCredentials()* ensure secure access control throughout the application.

The rationale behind selecting these classes lies in their direct correspondence to real-world entities and functional modules within the application domain. Each class adheres to the principle of encapsulation, bundling data with related operations, thereby minimizing external dependencies and facilitating isolated testing and modification.

Relationships among these classes include associations and dependencies that model interactions such as:

- **MobileNetworkUser** creating multiple **IssueReports** and generating **SignalLogs** through speed tests.
- **NetworkMetricsService** aggregating data from **SignalLog** and **IssueReport** instances to derive Quality of Experience analytics.
- **AuthService** providing authentication services to **MobileNetworkUser** and **SystemAdministrator** classes (the latter implied for administrative control).

By structuring the system in this manner, the class diagram not only clarifies the static organization of components but also lays the groundwork for robust implementation of interactions and data integrity. It empowers developers with precise definitions of each entity's responsibilities and interfaces, fostering cohesion and facilitating scalability in subsequent development phases.



Deployment Diagram

A *deployment diagram* is a crucial architectural model that illustrates the physical arrangement and configuration of hardware and software components within a system. It maps out how software artifacts are deployed on various nodes, including devices, servers, and networks. In system modelling, deployment diagrams provide a clear visualization of the system's physical infrastructure, enabling stakeholders to understand how components are distributed, interconnected, and executed in the real-world environment.

For the mobile network monitoring application, the deployment diagram encapsulates several key physical components and their relationships:

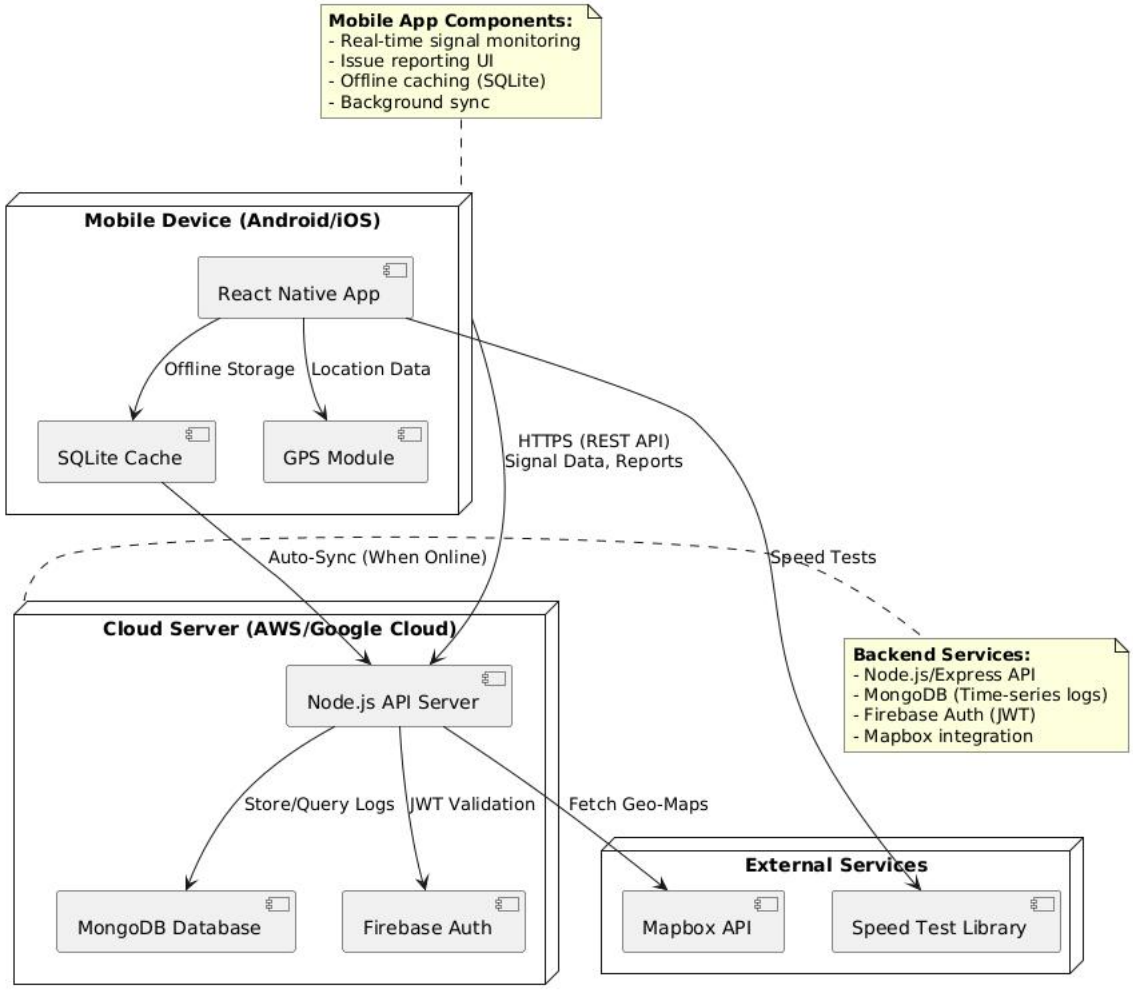
- **Mobile Devices:** The end-user devices (smartphones and tablets) where the application is installed and executed. These devices interact directly with users, collecting inputs and displaying results.
- **Cloud Servers:** Centralized servers hosted in cloud environments that manage backend services such as user authentication, data processing (e.g., speed test calculations), and network performance analytics.
- **Databases:** Persistent data stores located on secure cloud infrastructure used for storing user profiles, speed test records, issue reports, and system logs.
- **Network Infrastructure:** Communication networks—including mobile data, Wi-Fi, and internet connections—that facilitate data exchange between mobile devices and cloud servers, ensuring real-time responsiveness and data synchronization.

The diagram highlights several critical design decisions aimed at ensuring scalability, security, and performance:

- **Scalability:** By deploying backend components on cloud servers with elastic resources, the system can dynamically adjust to varying loads, supporting increasing numbers of concurrent users without degradation in service.
- **Security:** Communication between mobile devices and servers is protected using secure protocols such as HTTPS, with encryption safeguarding sensitive user data both in transit and at rest within databases. Additionally, firewalls and authentication services guard server environments.
- **Performance:** The distributed deployment reduces latency by utilizing Content Delivery Networks (CDNs) and regional server instances, ensuring swift access to application services regardless of user location. Caching mechanisms on mobile devices and servers optimize repeated data retrieval.

This physical architecture is integral to the system's functionality, as it defines the operational environment where software components execute, directly impacting reliability and user experience. By delineating deployment aspects, the diagram facilitates collaboration among infrastructure engineers and developers, informing configuration management, system maintenance, and future scaling.

Furthermore, the deployment diagram supports troubleshooting and enhances system documentation by explicitly showing dependencies and communication paths among hardware and software nodes. It also provides a reference model for compliance audits regarding data protection and network security standards.



Conclusion

Structured system modelling constitutes an indispensable foundation for the development and successful implementation of the mobile network monitoring application. Each diagrammatic artifact plays a distinct and complementary role in fostering a comprehensive understanding of the system's architecture and behaviour:

- **Context diagrams** delineate system boundaries and external interactions, establishing a clear scope at the outset.
- **Dataflow diagrams** map detailed data movements, ensuring clarity in process workflows and data management.
- **Use case diagrams** articulate functional requirements from the user's perspective, bridging stakeholder communication.
- **Sequence diagrams** capture temporal interactions and dynamic behaviour, guiding implementation logic and timing.
- **Class diagrams** define the static structure by modelling key entities and their relationships, facilitating modular design.
- **Deployment diagrams** illustrate the physical architecture, informing infrastructure decisions that impact performance and security.

Collectively, these models enhance communication between developers, administrators, and users, reduce ambiguity, and enable early detection of design issues. The thorough modelling phase provides a blueprint that supports efficient design choices, systematic testing, and maintainable code development. Ultimately, this disciplined approach underpins the delivery of a robust, scalable, and user-centered mobile network monitoring system aligned with academic standards and practical requirements.

References

This report references standard UML methodologies and academic texts on system modelling, including Booch et al.'s UML Guide (2005). Diagramming tools utilized include *draw.io*, *Lucidchart*, and *StarUML*, which facilitated the creation of the system diagrams. All sources are cited according to academic conventions.