NUS-ISS

MASTER OF TECHNOLOGY IN SOFTWARE ENGINEERING

Graduate Certificate Examination

Sample Examination Questions

Subject: *Architecting Scalable Systems*

APPENDIX A

'Multimodal Land Transportation Telemetry Platform' CASE STUDY

'Multimodal Land Transportation Telemetry Platform' Case Study

(The content of this case study is completely fictitious!)

The Land Transport Authority (LTA) envisions its next generation telemetry platform entitled Multimodal Land Transportation Telemetry Platform (MLTTP) that encompasses various modes of land transportation in Singapore. It aims to leverage on the latest innovations like 5G communication, cloud services, machine learning, business analytics, IoT and big data. The initial focus would be on the public bus transit network of the island and subsequently on the public rail transit network, public and private car hires, emergency services, etc.

Among the regular expectation of a well-engineered solution, the project on the public bus transit network emphasizes heavily on the timely collection and storage of data; low latency in the communication and dissemination of data; security of data (in particular sensitive user data); elasticity in supporting large number of concurrent users and transactions; real-time speed adjustment of buses in service; timely prediction of upcoming events (e.g. upcoming congestion in the next few hours); real-time adjustment of departing buses; as well as analysis of historical data for actionable insights (e.g. worsening passenger loading for a bus service over a stretch of its route during peak hours).

You are the Lead Solution Architect assigned to the project. You are responsible for the technical delivery of the platform that includes the architecture, design, development and deployment of the platform and two pilot applications in the first phase. After the first phase completes successfully, the platform should allow for expansion of new services and inclusion of new actors. The two pilot applications are a web application and a mobile application for the bus commuters.

1. Phase 1 of the Platform

The new platform supports the following actors:

1. Public User: Anyone may view the bus arrival times and passenger loads at an adhoc location.
2. Registered User: Anyone may register as a user of the platform. An authenticated Registered User may act as a Bus Commuter, Bus Captain or Data Analyst.
3. Bus Commuter: A Registered User who keeps a list of his/her favourite locations for viewing the bus arrival times and passenger loads.
4. System Admin: Member of LTA who administers the users of MLTTP.
5. Vehicle Onboard Unit: The telemetry device affixed to each bus handling communication with the platform, detection of vehicle location and speed, detection of passenger entrance/exit, etc. The Bus Captain manages his/her bus trip using the telemetry device as well.
6. Notification Gateway: The external system that helps to send email, SMS or push notification to a Registered User.

Figure 1 below illustrates the high-level functionality of Phase 1.

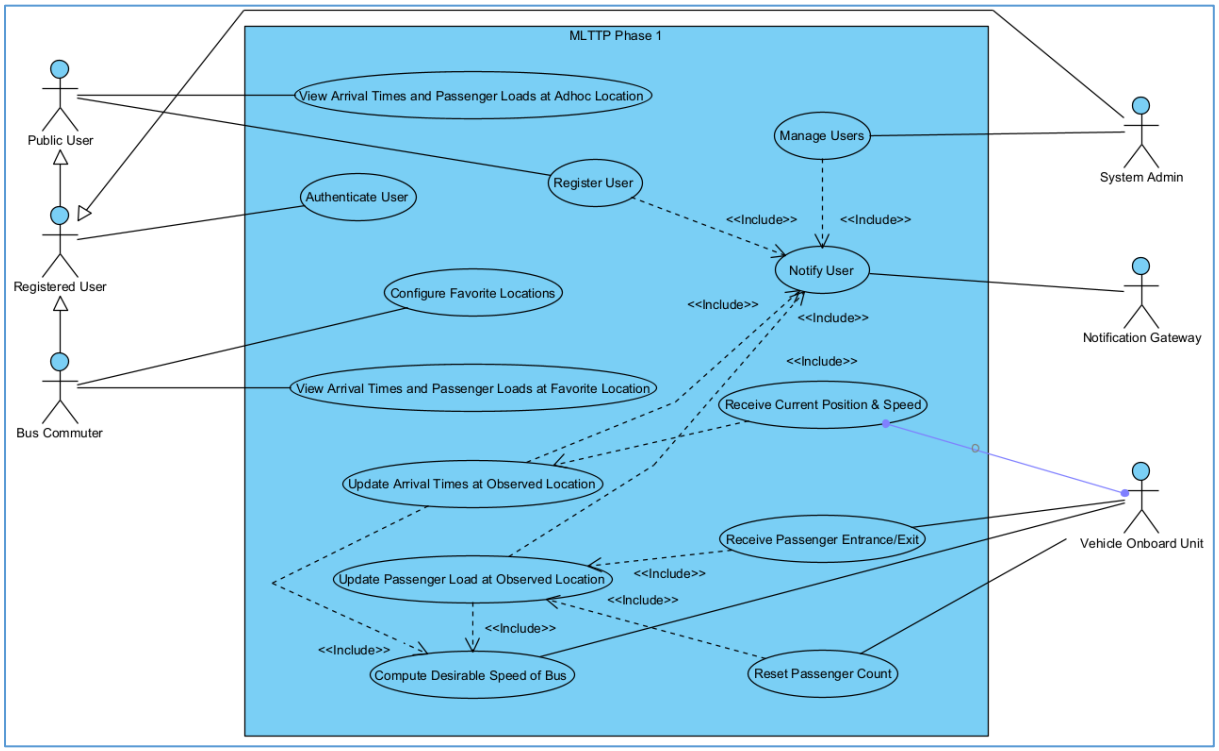

Figure 1: System Context for Phase 1

Here is a brief description of the use cases for Phase 1:

Use Case	Description
View Arrival Times and Passenger Loads at Adhoc Location	The Public User can view the bus arrival times and passenger loads of buses that serve a specific bus stop. The bus stop can be a bus stop near to the user location or a faraway bus stop picked from the bus service directory.
Register User	The Public User registers himself/herself with the system by providing his/her particulars. The platform verifies his/her identity by sending a verification email.
Authenticate User	The Registered User and System Admin logs in to the platform to start to use the platform features that require authentication.
Configure Favourite Locations	The Bus Commuter can configure his/her list of bus stops that he/she frequents.
View Arrival Times and Passenger Loads at Favourite Location	The Bus Commuter can view the bus arrival times and passenger loads of buses that serve a bus stop specified in his/her list of favourite bus stops. This is to cater to the locations that are frequented by the Bus Commuter.
Receive Current Position & Speed	Whenever the Vehicle Onboard Unit detects a discernable change in either the position or the speed of the bus it is affixed to, it updates the system with the new data.
Update Arrival Times at Observed Location	Whenever the system is updated with either the new position or the new speed of a bus, it computes the new arrival times of the bus at all the locations observed by the users via the Notify User use case.
Receive Passenger Entrance/Exit	Whenever the Vehicle Onboard Unit detects an entrance/exit of a passenger on/off the bus it is affixed to, it updates the system with the new data. For the expediency of update, it is intentionally designed so

	that the Vehicle Onboard Unit does not update the system in batches of entrances/exits.
Reset Passenger Count	As the passenger entrances/exits may not always be reliably detected due to commuter behaviours, the bus captain has to reset the passenger count at times. The Vehicle Onboard Unit would update the system whenever the passenger count is reset.
Update Passenger Load at Observed Location	Whenever the system is updated with either new passenger entrances/exits or new passenger count of a bus, it computes the passenger load of the bus at all the locations observed by the users via the Notify User use case.
Compute Desirable Speed of Bus	Whenever the system updates the arrival times and passenger load of a bus, it also computes the new desirable speed of the bus so as to optimize the overall waiting times of the all bus commuters in general. The new desirable speed is updated on the Vehicle Onboard Unit and the bus captain is expected to try his/her best to hit the target speed.
Notify User	The use case notifies the Registered User a message from the platform via email, SMS and/or push notification.
Manage Users	The System Admin administers the Registered Users of the platform. The platform notifies the Registered User of changes that affect him/her.

2. Phase 2 of the Platform

In addition to the Phase 1 functionality, this platform is being extended to support prediction of trip events and analysis of historical data.

There are two new actors:

1. Data Analyst: Member of LTA who administers the data analysis of MLTTP.
2. Bus Trip Planner: The system owned by the bus transit company that plans the departing times of new bus trips.

Figures 2 below illustrates the high-level functionality of Phase 2.

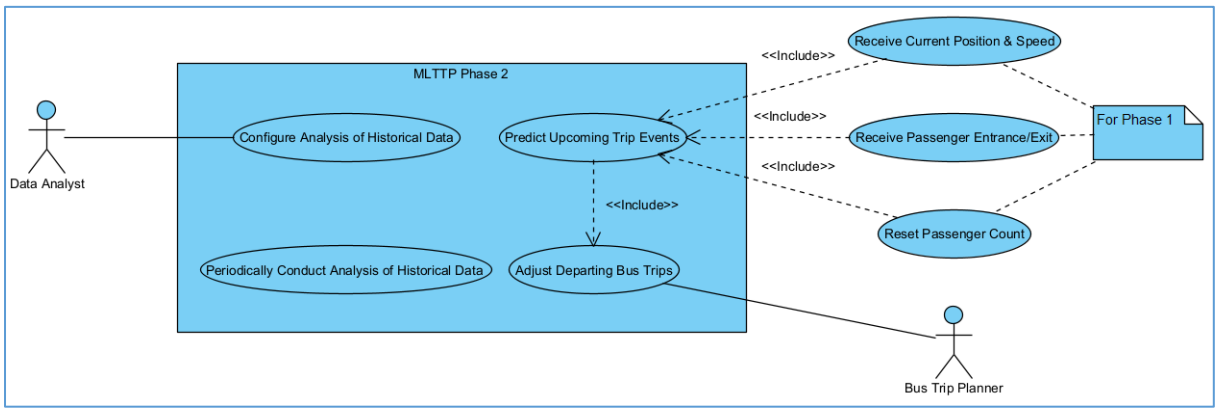

Figure 2: System Context for Phase 2

Here is a brief description of the use cases for Phase 2:

Use Case	Description
Configure Analysis of Historical Data	The Data Analyst configures the reports that are required for the platform. This can include the time of analysis, frequency of report

	generation, desirable actionable insights specific to the reports and parameters specific to the reports. E.g. worsening passenger loading for a bus service over a stretch of its route during peak hours.
Periodically Conduct Analysis of Historical Data	The system periodically conducts the analysis of historical data as configured by the Data Analyst.
Predict Upcoming Trip Events	Based on the new data received from the Vehicle Onboard Unit on position, speed, passenger entrances/exits and passenger counts of the buses, the system conduct machine learning to predict potential trip events in the next several hours. E.g. upcoming congestion in the next few hours.
Adjust Departing Bus Trips	Based on the potential trip events predicted by the Predict Upcoming Trip Events use case, the system computes the desirable departing frequencies for the affected buses. It updates the Bus Trip Planner system so that the latter can optimize the departing times of new bus trips accordingly.

3. Extensibility of the Platform

The platform is designed to use common services and libraries that support and speed up the development of future services provided by the platform.

4. Performance and Scalability of the Platform

The platform is designed for the following performance and scalability:

concurrency
- For phases 1 and 2, the platform supports up to 100,000 concurrent Public Users, 10,000 concurrent Registered Users, 5 concurrent System Admins and 10,000 concurrent Vehicle Onboard Units.

latency
- For interactive request-response type of operation (e.g. View Arrival Times and Passenger Loads at Adhoc Location), the expected response time is less than 3 seconds for 90% of the requests.
- The platform is not expected to handle multitenancy.
- The platform handles increasing amount of data while maintaining the response time of the application.

concurrency
- The platform should cater to up to 3,000 buses as the fleets are expanded.

latency
- The platform should process updates from buses near real-time. For Update Arrival Times at Observed Location and Update Passenger Loads at Observed Location, changes are expected to be shown to the users within 5 seconds. For Compute Desirable Speed of Bus, changes are expected to be sent to the Vehicle Onboard Unit within 2 seconds.
- The platform should ensure isolation of data without sacrificing the consistency of the update and retrieval of data processed in real-time.

5. Security Architecture of the Platform

The platform is designed with the following security considerations:
- The system follows general security best practices in infrastructure and application security e.g. reverse proxy in DMZ, etc.
- Personal information is stored in encrypted form and the platform is able to find all information stored about an individual. According to GDPR, the actors should allow

users to request access to user's data and the platform should be able to facilitate the compliance.

- Sensitive tenant information is encrypted with tenant-specific encryption key to avoid accidental data leakage due to software bugs. The encryption keys are stored securely and the system is able to handle change of encryption key through an automated process.
- Subsystems that handle payment and cardholder information comply with PCI-DSS and therefore payment and cardholder information handling is limited to as minimal subsystems as possible to minimize cost and minimize risk exposure.

6. Initial Architecture of the Platform

As the Lead Solution Architect, you have provided the following architectural guidelines for the platform and the applications.

6.1 Architecture Layers

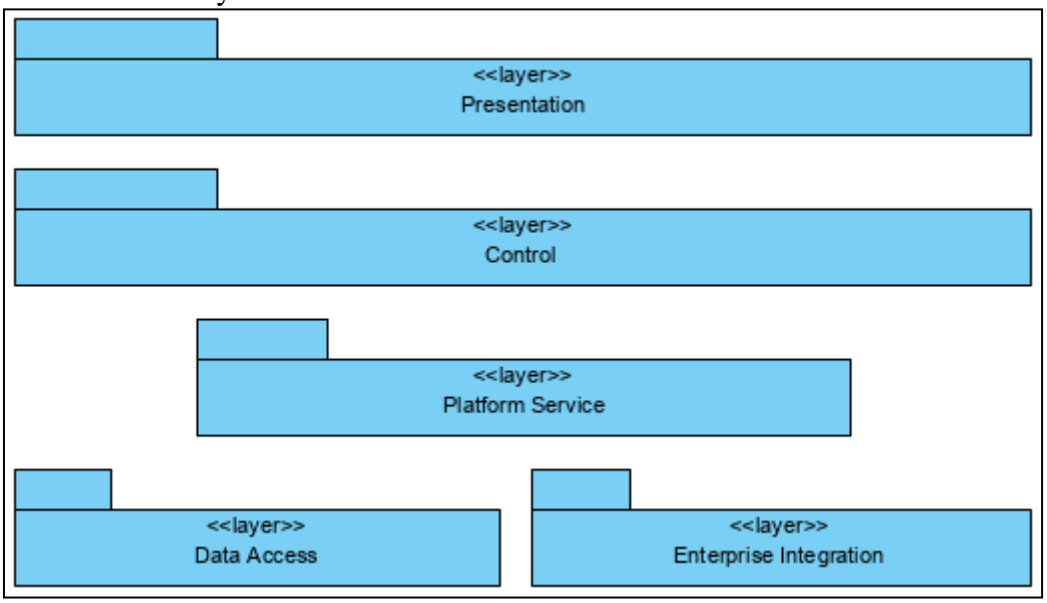

Figure 3: Architecture Layers

The above layered architecture illustrates the desirable layering of a typical application on the platform. The Presentation layer handles the user interaction of the application. The Control layer handles the flow logics of the application use cases. The Platform Service layer hosts the shared/reusable services. The Data Access layer handles the access to persistent storages. The Enterprise Integration layer handles the integration with external systems.

6.2 Logical Architecture Overview

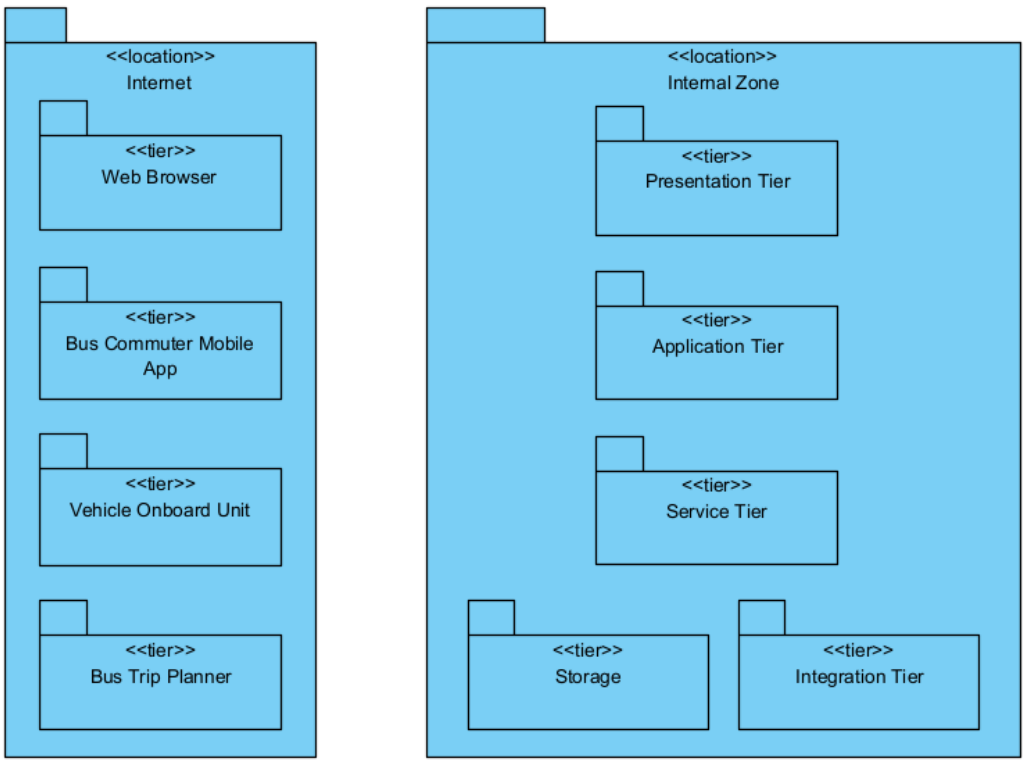

Figure 4: Architecture Overview

The above logical architecture overview illustrates the desirable tiering of the platform. The frontend tiers are in the form of Web Browser (for the Bus Commuters and System Admins), Bus Commuter Mobile App (for Bus Commuters), Vehicle Onboard Unit (for bus captains) and Bus Trip Planner (belong to bus transit companies). The backend tiers are Presentation, Application, Service, Storage and Integration.

6.3 Deployment Options

The platform is to be designed as a cloud native solution. The services should be implemented as microservices.

7. Domain Model for Phase 1

A Software Architect has analyzed the requirements for Phase 1 and identified the following domain entities, services and aggregates. Note that the aggregates are marked using irregular shapes in red. The Development team would implement the platform according to this domain model.

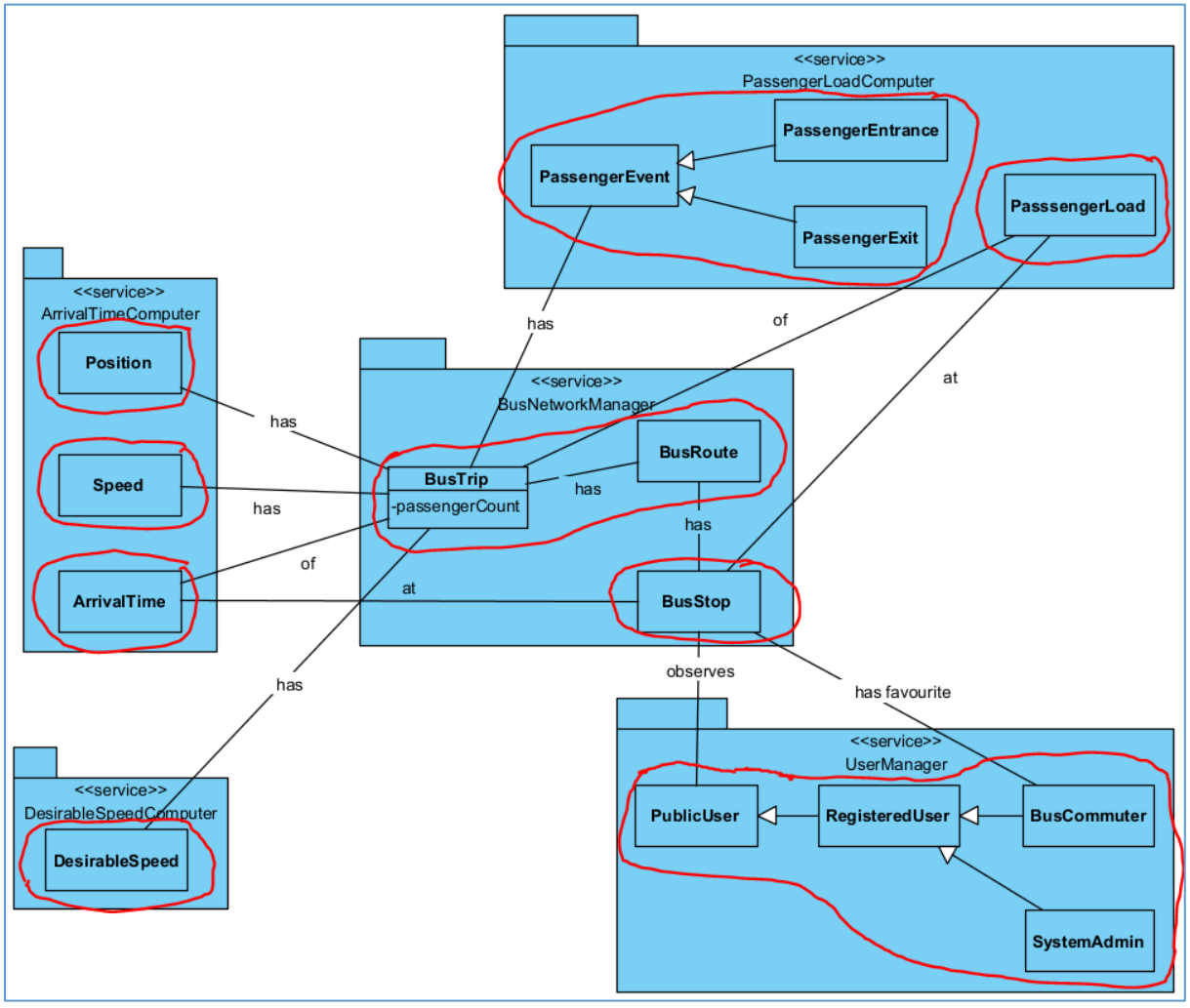

Figure 5: Domain Model for Phase 1

8. Logical Architecture for Phase 2

A beginner Solution Architect has devised an **incomplete** logical architecture for the requirements for Phase 2. The emphasis is on the packaging of the functional elements of the use cases into subsystems. When it is eventually completed, the Development team would implement the platform according to this architecture. Figure 6 shows the logical detail deployment diagram (note: the deployment nodes are not shown) for the use cases.

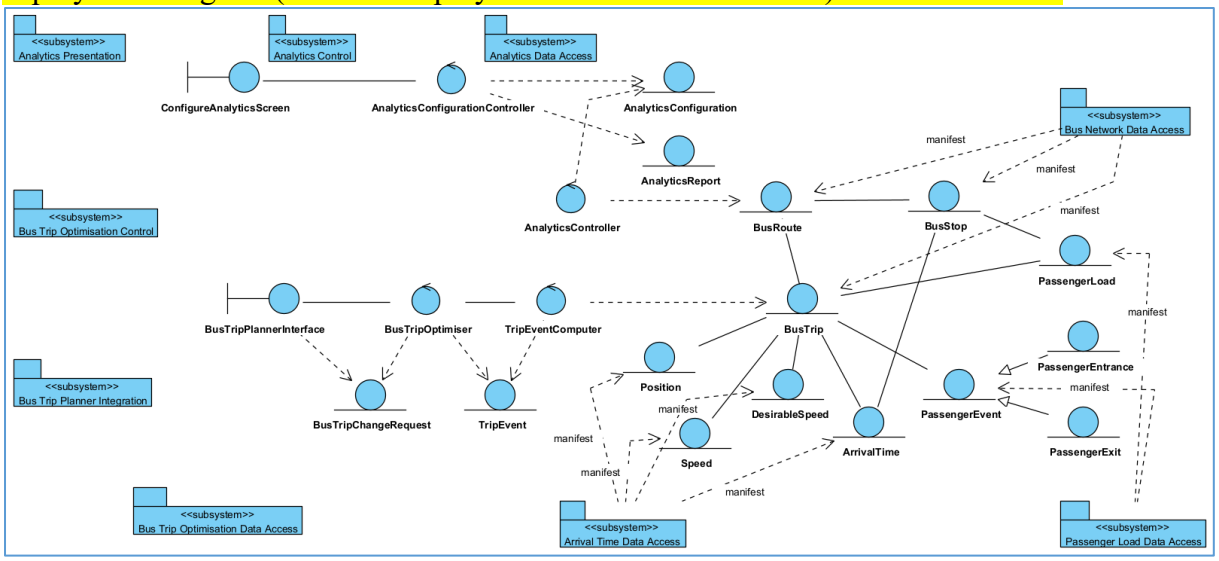

Figure 6: Logical Architecture for Phase 2

In the context of solution architecture, a " manifest" typically refers to a document or artifact that outlines key aspects of a solution or system. This document serves as a clear and concise representation of various elements of the solution, helping stakeholders understand its purpose, components, and design principles.