CS 685 Spring 2017

Internet Of Things – Car's Ecosystem

Milestone 3

Car Fleet Management System

Team 2

Anthony Picaro – arp223@njit.edu

Nainika Aleti – na368@njit.edu

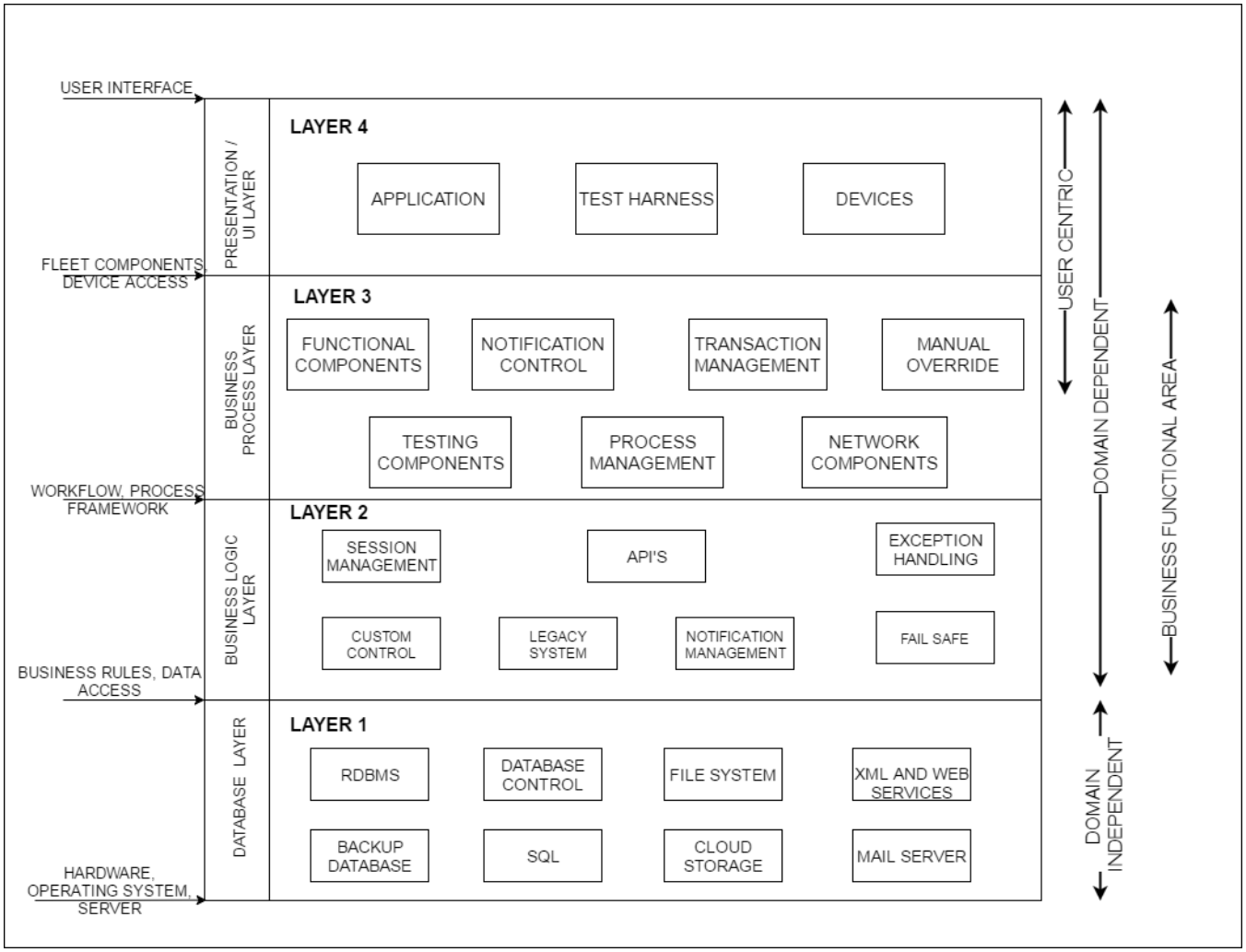
Vivek Bharathwaj Ravikumar – vr329@njit.edu

1. Introduction

In the third milestone, the creation of the Development View Architecture became the focus of the project. The first task was to draw the layers for the Development View; the rest of the milestone depends on the layers, so this was the main priority. Once this was done, each layer had to be described with respect to their responsibilities. Upon completion of the layer descriptions, we were then able to create modules and add them to a separate copy of the Development View drawing, along with briefly describing each module's purpose and functionality.

The modules are not useful if they do not align with the Process Architecture, so relation between the newly created modules and the Process Architecture had to be established and explained. Lastly, but most importantly, the Development View is useless if it has no bearing to the Reference Architecture, so their relation had to be explained, including all assumptions made about the Reference Architecture in order for the Development View to work as intended, along with service requirements or assumptions.

2. Development View Diagram



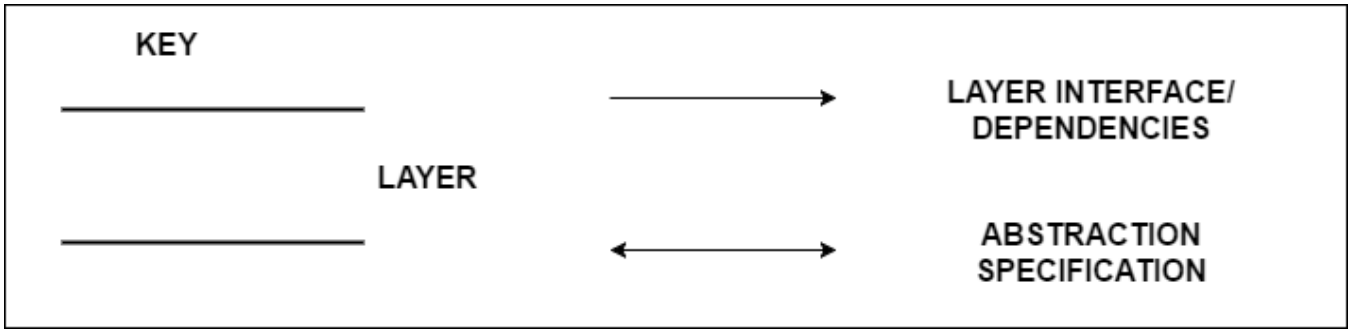


Diagram 1 – Development View

The Development View Diagram has 4 layers:

1. PRESENTATION/ USER INTERFACE LAYER
2. BUSINESS PROCESS LAYER
3. BUSINESS LOGIC LAYER
4. DATABASE LAYER

Each of these layers in the architecture provides some functionalities to the application that are specific to those layers. These layers are classified based on the main idea of “Separation of Concerns”, allowing the components and modules that are present in each layer to focus on their own functionality rather than concerning themselves about what is happening in the adjacent layers. By doing this, the software provides the same level of service in multiple layers as if everything was accomplished in a single layer. In addition, these components and modules could be easily split among the development teams, helping with maintenance of the application without any hassle. This Architecture closely resembles the 4-Tier Deployment, which is done for application security.

2.1 Presentation/ User Interface Layer

This is the top-most layer and is what is visible to the end-user. This layer is responsible for providing a User Interface to the end-user, through which they can access the application. Because the Application is developed for use in cross cutting platforms, like Web Application (Chrome, Firefox, IE, Safari), Mobile Application (Android, iOS, Windows), and Desktop Application (Windows, Unix/Linux, Mac OS), this layer may use the services of all of the lower layers, such as HTTP, network components, APIs, and the databases. Strictly speaking, lower layers cannot access this layer, leading to this layer to be platform and programming language independent.

2.2 Business Process Layer

The Business Process layer has all the components that contribute to the functional attributes of the application. Business process is made of several functional components that contribute to the brain of the application, focusing on what the application should do. The Presentation/User Interface Layer is responsible for displaying the components of this layer using Service Oriented Architecture. Most of the code for the functionality of the application is contained in this layer and this code should be independent and loosely coupled. The bottom line being the Business Process Layer plays a vital role in orchestrating the interaction between business level requirements and technological solutions.

2.3 Business Logic Layer

All Business related logic is implemented in the business logic layer, with all components in this layer strictly adhering to the business rules whenever it has to process a request. It also interacts directly with the database layer to read, write and update the appropriate file. The design pattern that is most commonly used in an application is if the Business logic layer is called from the presentation/UI layer, then the components are servicing a request from the user. Whereas if the same layer is called by the business process layer, then the components are being used as a part of process definition.

2.4 Database Layer

The components in this layer can send and receive data from other layers. They also have the means to store this data and access it via a File System. This layer uses a Relational Database Management System (RDBMS), which hides the complexity of its database interaction.

2.5 Design Rules.

1. A Top-Down Approach

Layers that are higher-up in the Development View are able to access lower layers, but the reverse is not true. A primary example of this is the Presentation/User Interface Layer; it can access all of the other layers, but none of the other layers can access the Presentation/User Interface layer. Likewise, the Business Logic layer can access the Database Layer, but cannot access the Business Process layer.

2. Layer Abstraction

This Architecture facilitates Layer Abstraction, wherein the implementation details about the process and functionalities are hidden from the end-user. This separation of concerns about the layer helps achieve greater platform independence and interoperability.

3. Reusability

Because of the layered approach, subsystem and modules are identified in each layer based on the functionality of the respective layer, helping to achieve a certain level of reusability. For example, whenever the presentation/UI layer calls a component in the business logic layer, it serves the user request, but when the business process layer invokes the same component, it acts a part of process definition.

4. Security Compliant

The security is provided by the existing Internet of Things system, having the ability to define policies, enforce these policies, and verify the compliance of the elements of process with a set of predefined policies.

5. Layer Interface/ Dependencies

The Database layer is built upon the Operating System and hardware. The Business Logic Layer invokes the database layer using business rules for data access. The Process Framework provides a method of workflow and

for the interaction between Business Process and Business Logic Layer. The Presentation layer access the fleet components in the business process layer and displays them using User Interface.

3. View Alignment with Reference Architecture

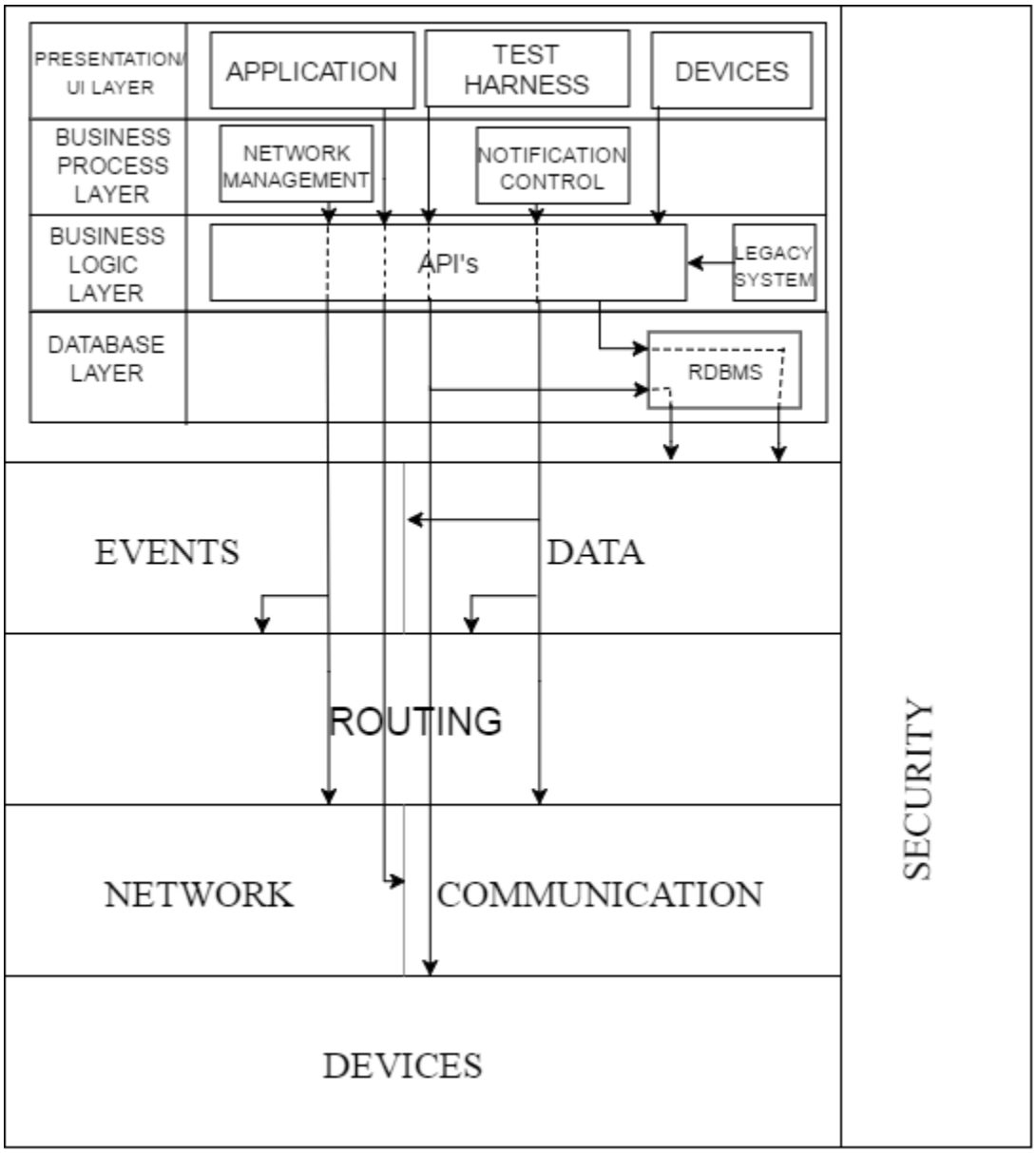


Diagram 2 – View Alignment with Reference Architecture

3.1 Why These Layer Matter?

Presentation/UI Layer

The reference architecture requires that there be a user interface that is compatible with any given off-the-shelf consumer device that has a visual display and user input; the Presentation/UI layer

focuses solely on that. The devices are detected on this layer and the appropriate graphic user interface is displayed based on the type of device. This layer also contains the OBD port dongle, which polls the vehicles for relevant information and sends the information to the databases, which is then shown on the graphic user interface. This layer of the Development view corresponds to the bottom layer of the reference architecture, specifically the devices and OBD port dongle. It also is what the user would end up referring to as “the fleet management software”, so the Presentation/UI layer could also be considered the Fleet Management layer.

Business Process Layer

The Business Process layer is exactly what the name of the layer implies and contains the most important part of the reference architecture: the services provided to the users. All of the functional components are included in this layer, such as the telematics tracking and vehicle location monitoring, and all of the network components are enveloped in this layer too. Without this, the software would have no sense of what information to track and report, nor would it have a method to distribute the information.

Business Logic Layer

The Business logic layer provides the Application Programming Interfaces (API’s) that are required by the Presentation/ UI layer and the Business Process Layer’s module or subsystem to interact with the services provided by the IoT. These layers access those underlying services using the API provided by the Business Logic Layer

Database Layer

The reference architecture specifically mentions “Database Control”, which requires the databases to exist and the ability to use and monitor them, which is what the Database layer in the Development view is.

When the users of the fleet management software sign-up, they expect the information about their vehicles and routes to be recorded in real-time for future reference; the database layer’s main database exists for that purpose. The database layer contains a main database and backup database(s), as well as a cloud storage option, to provide multiple copies of the same information; if one storage option fails, the others are unaffected and can be used in the failed one’s place.

4. Modules in Development View

The following diagram displays the development view after the modules and subsystems are inserted in the appropriate layer. The modules/ subsystem in one layer interacts with themselves and with the modules/ subsystem in the bottom layer in order to provide an overall functioning architecture

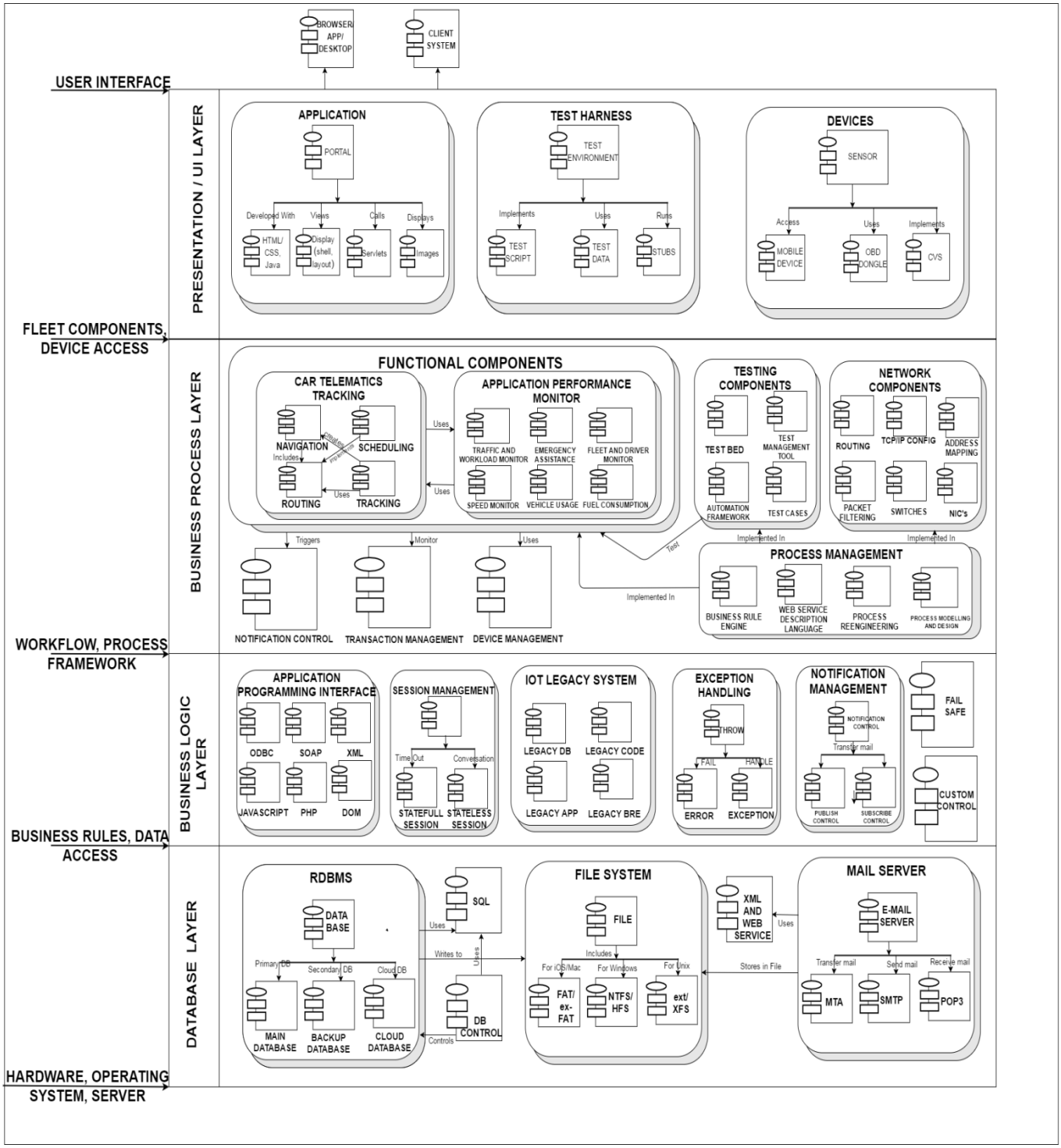
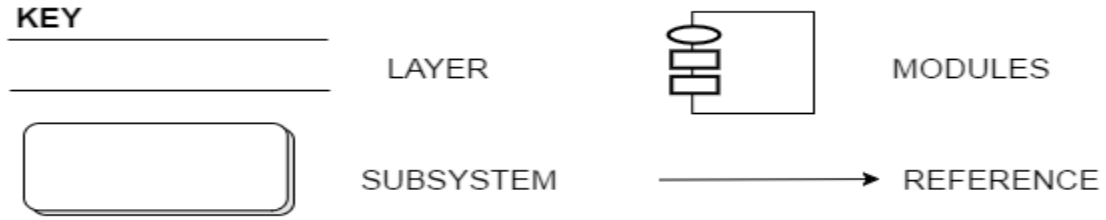


Diagram 3 – Modules And Subsystem in a Layer



4.1 Modules/Sub-system in Presentation/ UI Layer

Application – The entire User Interface of the Application is present in this layer and acts as a portal through which the end-user can access the application. The subsystem and modules in the business process and business logic layer are displayed in the front end using this interface. It is comprised of a portal that is developed using HTML, CSS, and Java. This portal is in turn displayed in the GUI across several platforms like browsers, mobile app or in desktop.

Test Harness – The Test Harness provides a separate environment within the Organization where the Quality Assurance team can access a specified version of the application and test it under different conditions with several hardware, software and run-time constraints. Test Scripts are run on the test environments.

Devices – All the devices that provides a display portal to the end user reside in this layer as they have the Graphical User Interface methods that can display the output.

4.2 Modules/Sub-system in Business Process Layer

The following are the list of components present in the Business Process Layer that interact with each other and with the other layers present in the architecture.

Functional Components – The Functional Components includes the Architecturally Significant Requirements that were identified in Milestone 1; these components decide what functionalities the application should provide. A few of the components are Car Telematics Tracking, Fuel and Speed Tracking, Fleet and Driver Monitor, Vehicle Usage Analysis, Traffic and Workload monitor and emergency assistance.

Testing Components – The Testing Components includes all the processes like the test bed, test cases, and the test management tool that are required for the test harness to test the application. All the test cases are executable using a test management tool, like HP Application Lifecycle Manager (HP ALM), where the lifecycle of an application is trackable. This also helps in monitoring the defect status until it is resolved.

Network Components – The Network Components provide methods to the functional components in this layer and to the devices in the higher layer, so that all vehicles and devices can connect to the servers and the user can access the application that requires network related functionalities without any problem. For example, scheduling a cab requires location monitoring and GPS, while accessing the application requires an internet connection.

Notification Control – Notification control accesses the Notification Management from the business logic layer and helps to trigger/push the notifications, such as notifying the passenger about the vehicle's arrival and notifying the manager of any event that occurs.

Process Management – The core concept/functionality of process management is to bind and assemble the communication between different business logic components. It also performs the process planning and business process performance monitoring. The Business Rule Engine present in the Process Management provides the basis for a business policy and procedure that the architecture must adhere to.

Transaction Management – Because the app uses different IT services across several layers, it would be helpful to have a tool that provides a system for managing technological related services from business perspective, which is where the Transaction Management module is introduced. Some examples of transaction management would be finding and warning about an unexpected system overload, time of arrival of a message or receiving time of a message, and even a transaction in terms of payment for service used or a payroll system. The underlying goal of this is to improve the Quality of Service.

Manual Override – Since most of the components in this layer are automated, a manual override is introduced in the event of a failure; if the computer itself cannot respond or rectify the failure, the manual override can be implemented to rectify or respond to the issue in place of the computer, like a Denial of Service Attack on the system.

4.3 Modules/Sub-system in Business Logic Layer

Application Programming Interface - The Application Programming Interface module is the integrated development environment that is designed specifically to create and maintain the application.

Session Management - The Session Management module determines whether the connection between the server and each client is stateful or stateless.

IOT Legacy System - With every software project, it is inevitable that the software will receive updates throughout its lifecycle to guarantee the best possible experience for the users. If a user is going to use an older version of the application when an optional update is available, the legacy application must still work as intended.

Exception Handling - The exception handling module is exactly what the name implies; should and exception be 'thrown', the software must have the ability to determine if the exception is indeed an exception or an error.

Notification Management - Notification Management handles the publish and subscribe which ensures the timely delivery of published information to the appropriate person. The Publish module allows a process to publish a message using a broker. This allows a process to send the message to another process and it has the capability to initiate an asynchronous communication. The Subscribe module allows a process to wait for the message and deliver it to the particular process.

Failsafe - Should an issue arise in the software, there has to be something that can handle the issues and react accordingly to minimize the potential impact.

Custom Control – This is a Java control containing reusable functionality; this control is extensible and can be used to call other Java controls. It also supports asynchronous communication.

4.4 Modules/Sub-system in Database Layer

RDBMS Module - The server will have more than just one database to contain all of the information related to the fleet management software, so a database management system has to be implemented.

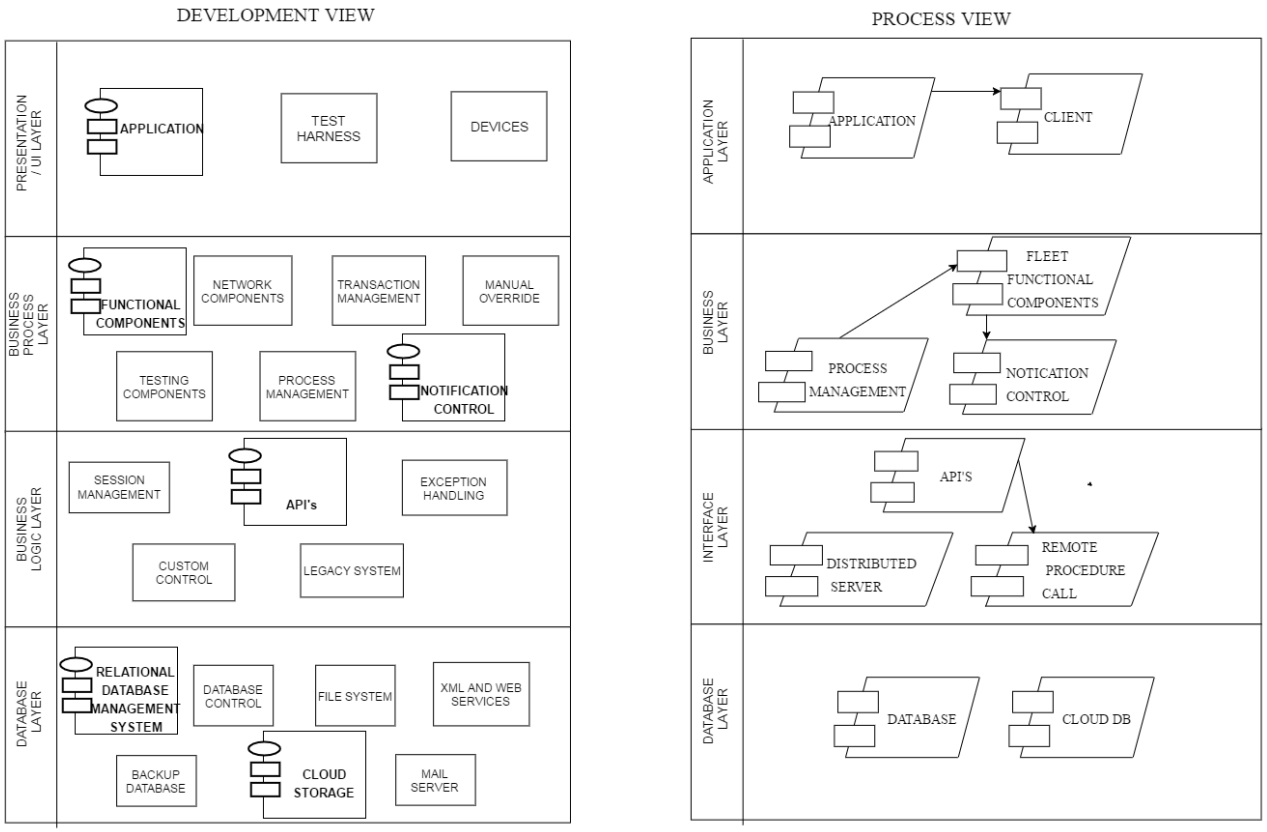
SQL Module - The database will use SQL, but SQL is not the only programming language used to operate databases, so it is a module because another language can be substituted in at any given moment without affecting the databases.

Database Control Module - There has to be a way to monitor and control the operation of the databases, so a Database Control module is introduced.

File System Module - The File System module handles the different possible file systems encountered within the database layer, such as FAT and NTFS.

Mail Server Module - With any software or website, or even any company, there will be mail sent to and from the clients, especially so when a support center is involved. A Mail Server has to be implemented to handle the mail.

5. Alignment with Process View



The Development View contains four layers: the Presentation and User Interface layer, the Business Process layer, the Business Logic layer, and the Database layer. Each of the layers in the Development View encompass modules that provide various functionalities to our application depending on which layer they belong to. Similarly, the Process View architecture can be broken down into four different layers: the Application layer, the Interface layer, the Business layer, and the Database layer. Each layer in the Process View holds several components that interact with other components in the same layer and with other layers in order to yield better performance.

The Presentation layer in the Development View contains the components that implement and display the user interface and manage user interaction with our application. This layer includes controls for user input and display. Similarly, the Application layer in the Process View provides a lot of the interface that runs through the body of the application to provide swift interaction between the front end, back end, process, and events.

The Business Process layer in the Development View includes information exchange flow and coordination between the processes, individual users, and the application to monitor performance and achieve the goal of fleet management. The Business Layer in the Process View implements all the Functional Requirements in the form of processes; These process communicate with each other using an Inter-Process Communication Interface, where it interacts along with other process and process interface and provides the necessary information to the front end (GUI).

The Business Logic layer in the Development View contains logic specific to the business domain, which can be used by other applications as well, such as a set of web services exposing a well-defined API. The Interface layer in the Process View acts as middleware between the application and the process, over which they communicate application and business operations. The Database layer in the development view contains components that interact with the rest of the application to store, retrieve and backup fleet related information.

Similarly the Database layer in the process view contains components such as RDBMS and cloud storage that facilitate storage and access of data, which can be recovered in the event of a disaster recovery operation.