The Titan Smart Navigation System

Software Architecture Document

By Titanscrum

Annamalai Arumugam

Benjarit Hotrabhavananda

Hetal Patel

Keshav Lingala

Kishore Shankar Abimanyu

Nitish Lele

Thomas Tran

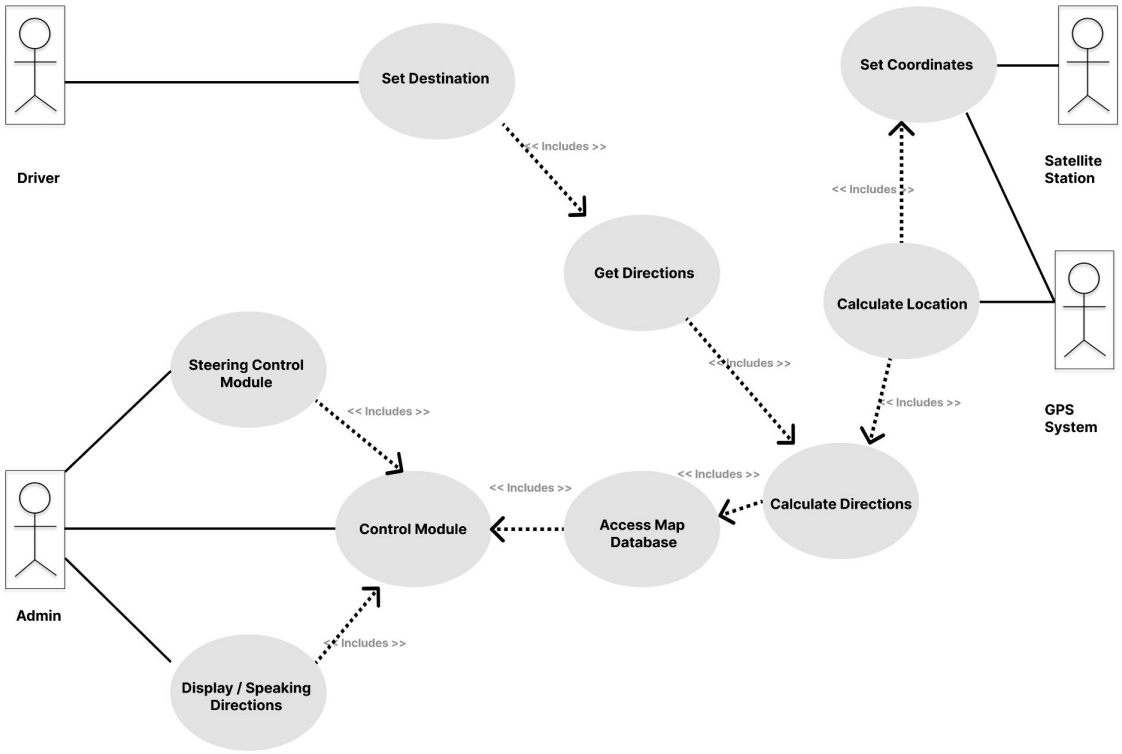
Vinay Shah

Introduction

- The Smart Titan Navigation and Steering System in which the entire automatic operation of the car is described by taking all feasible methods into account.
- We looked into various techniques to get a general architecture of how the process works and how it can be fine tuned to address flaws in existing systems.
- The architecture build is developed by keeping all these major purposes ,scope in mind for long term so the output generated will be very appropriate and effective.

Use Case Diagram

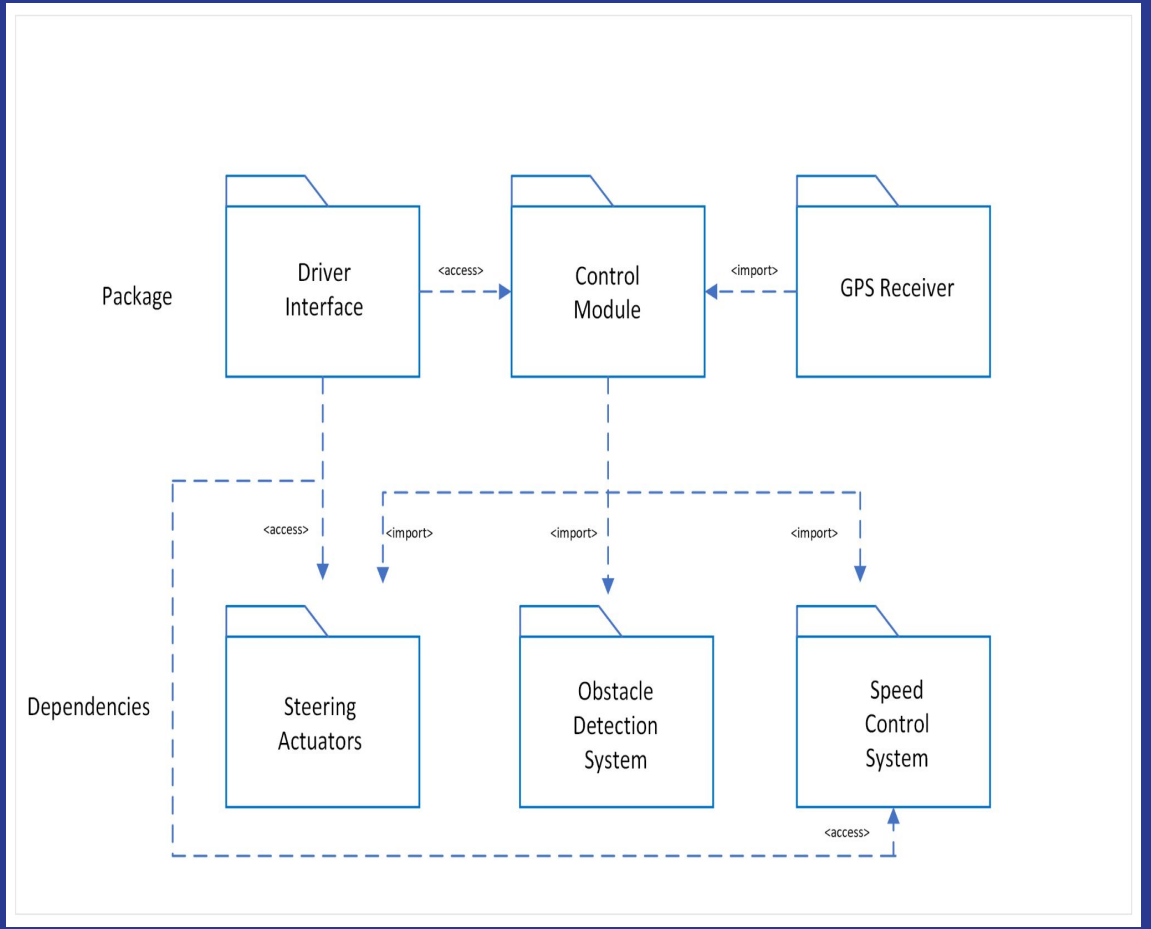
- The driver Sets the destination in the System(STNSS).
- The System takes in destination input, gets directions, calculates the location, gets coordinates from GPS, and then accesses the map database for additional information.
- Then, The System will pass all the Data about the maps and directions to the central control module, which then displays it on the computer screen and sends it to the steering control module.
- The Steering Control Module then Navigates the Car.



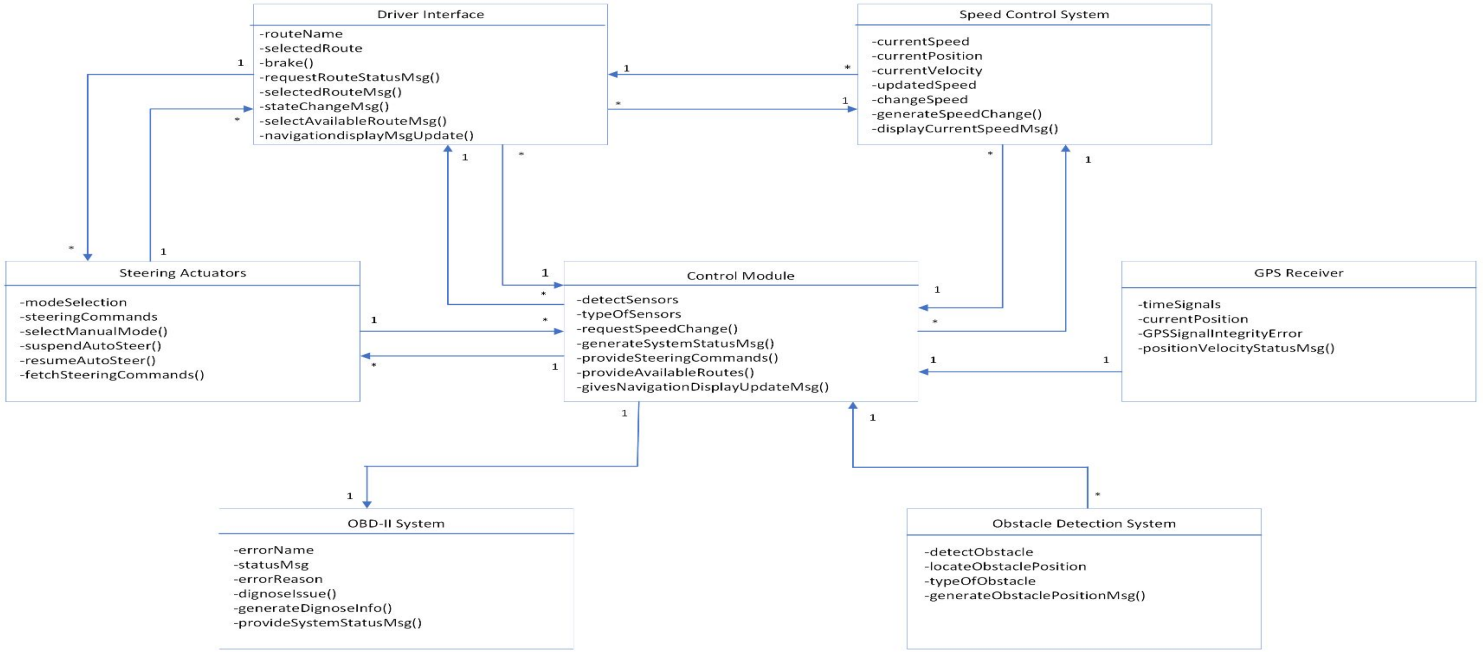
Logical View

- Package:
  - Driver Interface
  - Control Module
  - GPS Receiver

- Dependencies:
  - Speed Control System
  - Steering Actuators
  - Obstacle Detection System
  - OBD-II System

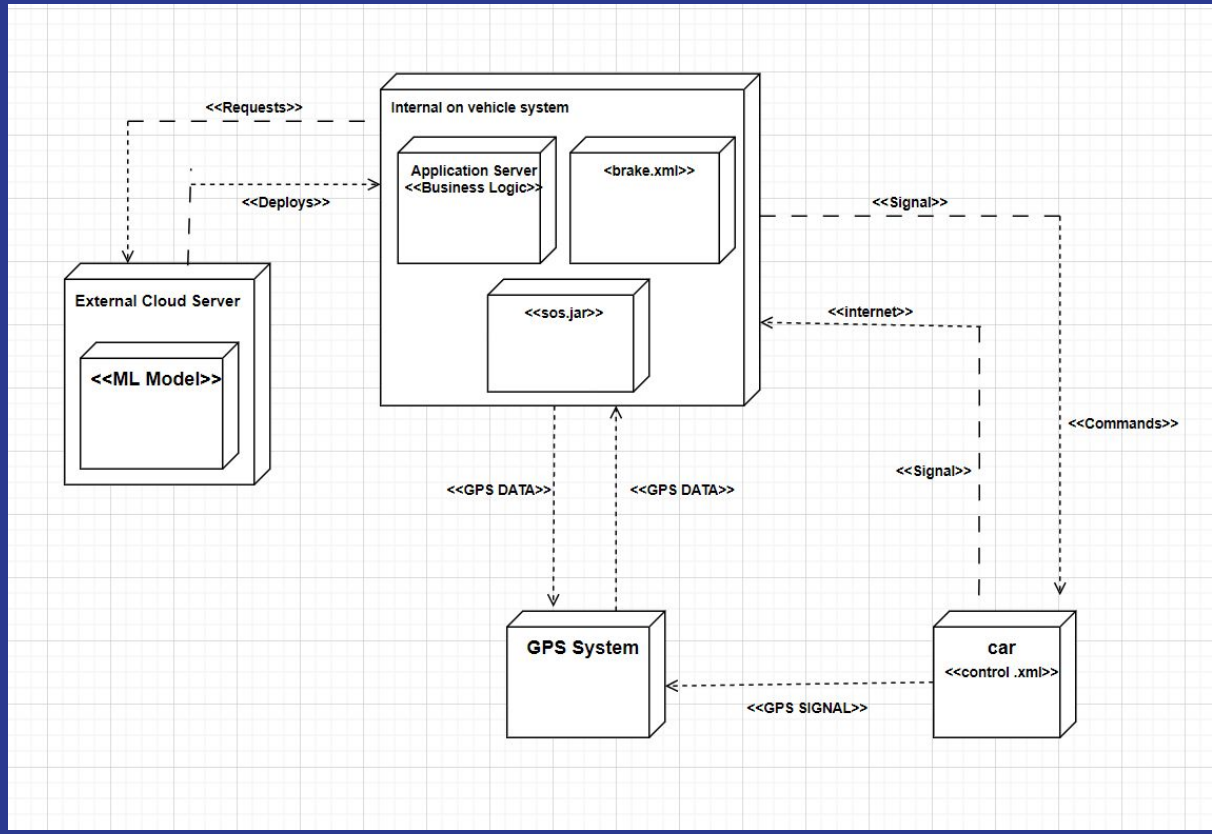


Logical View



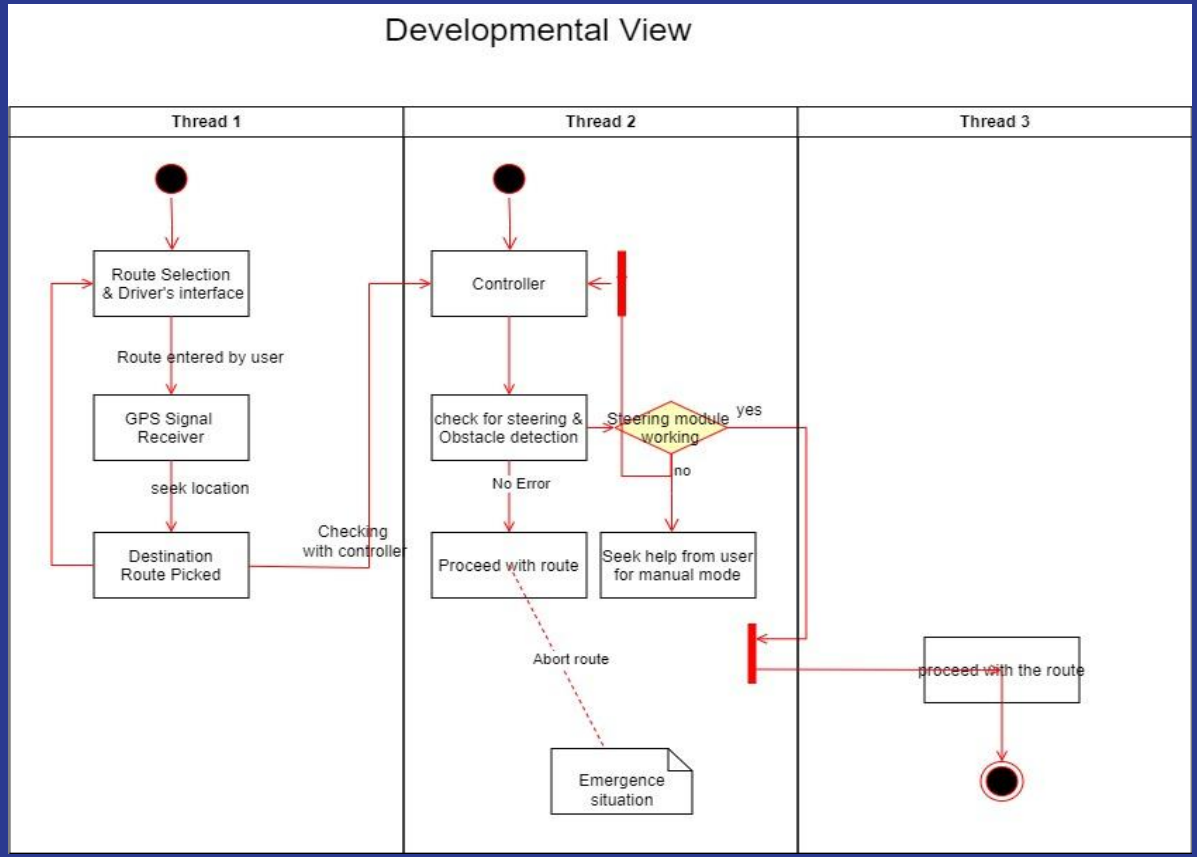
Physical View

- GPS System
- Control system(For steering control)
- Emergency SOS system
- nternal on vehicle System
- External Server / Cloud storage



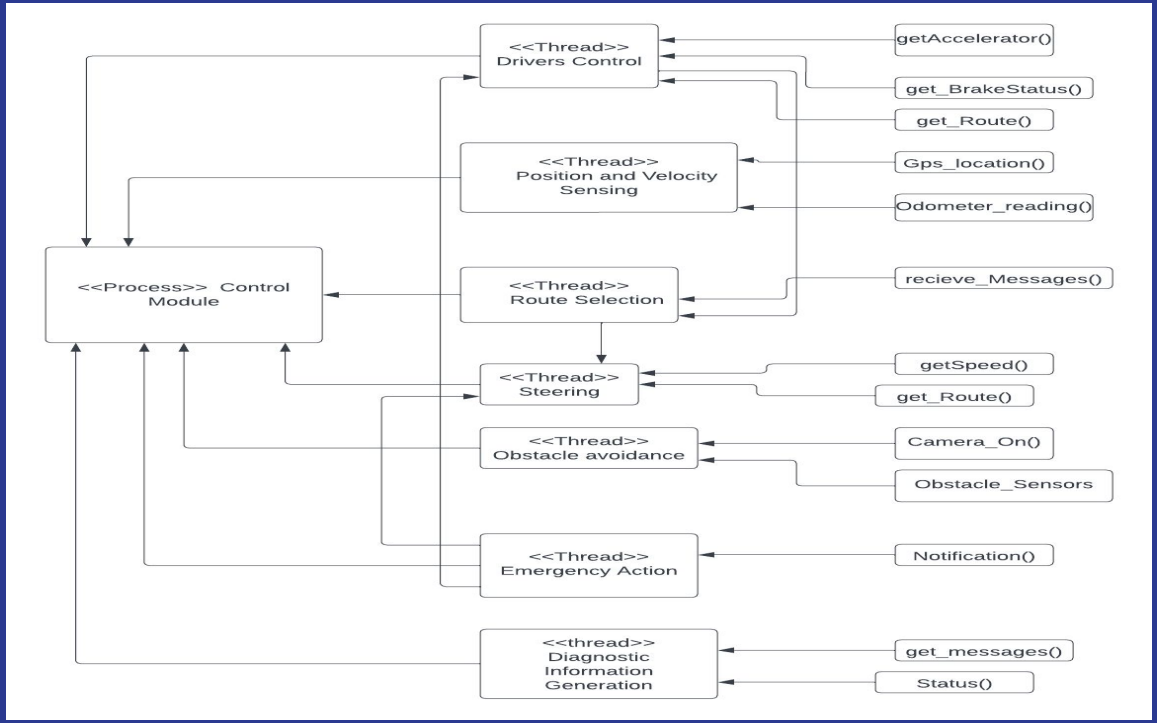
Developmental View

- Multithreaded application.
- Route selection thread takes GPS signal by seeking location from GPS process running in background.
- Once destination route is selected, it pushes to controller thread.
- Controller then takes appropriate actions with steering and emergency situation module.
- Finally, the last thread which actively proceeds to the route.



Process View

- Control Module acts as a central processing unit.
- There are several threads that are connected to control module.
- Each thread has its own functions that can be performed when called.
- Eg: Position and Velocity sensing has odometer_reading() function.



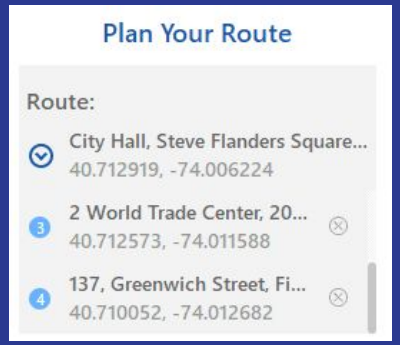
Route Selection - ASRs

- The component shall be able to **accept routing requests** when necessary

- The component shall be able to **identify up to three possible routes** if any routes are accessible for given specific destination.

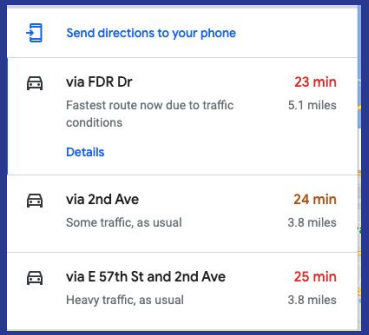
- The component shall allow the driver to **choose the desired route to the destination** from a list of available options.

Route Selection - Detail Use Cases

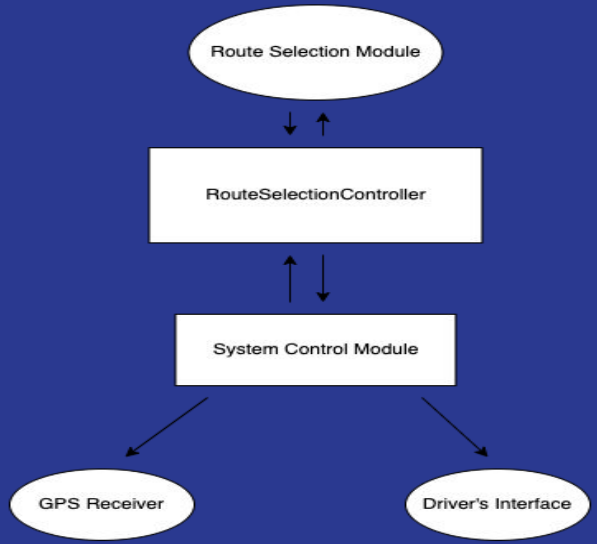


- The component shall be able to **receive routing requests** when necessary
    a. The driver shall be able to enter the destination address in either latitude and longitude format or human-readable form.
    b. The driver shall be notified of when the routing request is not accepted by the system.
    c. The component shall be able to **identify up to three possible routes** if any routes are accessible for given specific destination
        i. A list of possible routes to the specified destination will be shown to the driver
        ii. The driver shall be notified if the module is unable to determine a route to the specified destination

Route Selection - Detail Use Cases (Cont.)



- The component shall allow the driver to **choose the desired route to the destination** from a list of available options
    - The component shall only accept one route from the driver.
    - The system shall be notified when the driver make a selection



Route Selection - Logical Architecture

- Input data will consist of
  - Position Velocity Message
  - Routing Request Message
  - Selected Route Message
- Output Data will consist of
  - Available Routes Message
- The destination address selected by the user will be entered into the system through the **Driver's Interface**, where it will be passed on to the Route Selection module for further processing
- The module will determine the best route utilizing the data that has been provided by **other components within the system**.
- The module will have a **component for route computation**.

Route Selection - Physical Architecture

- The module will run under **Linux operating system** as a standalone process to accelerate execution speed.
- The component shall have a **mechanism for "listening" to system events** and acting appropriately.
    - Eg. the System Status Message indicating that the new route is required from the driver therefore the old one is not longer valid.
- The route to the destination will be calculated by the module using the data from the **GPS receiver** and **map**
- The module will **provide the specific route informatio**n to the system for steering capability.
- The module must provide **two-way communication** so that the system can alter attributes and request certain actions be taken.
- A **route selection controller** will be present in the system that manages all routing-related functionality and state variables as well as communication with other system components

Route Selection - Physical Architecture

- What are the architecture decisions for this component?
  - Publish/subscribe
    - System Status Message
  - Layered pattern architecture (MVC)
    - Route Selection Controller
    - Middleware

Concept of Operation

Driver Control:

- The system of the car will comprise of two options available for the user which is **Suspend** and the other is **Resume**.
- When the driver prefers to select manual mode message the system should suspend automatic steering functionality and let the driver take over the control for driving.

Positions & Velocity Sensing:

- Automatically Update its current Position and Velocity periodically.
- Automatic Disable automatic steering functionality when there is a sensing information is not appropriate.
- Also, If there is an GPS Signal Integrity Error within 2 seconds, disable automatic steering functionality.

Concept of Operation

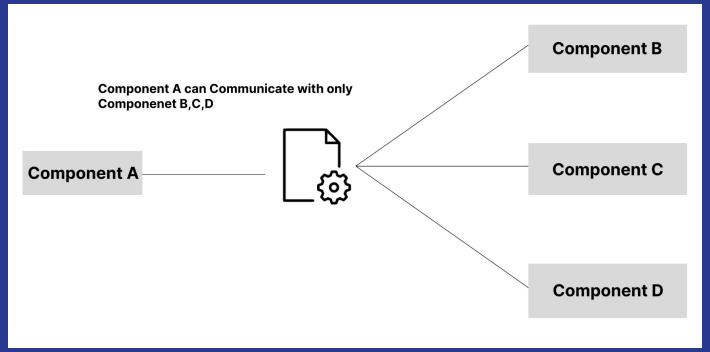
Steering :

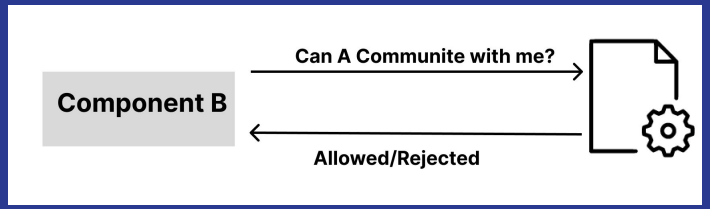
- Actuators control the steering when the car is in automatic mode.
- The control module determines the angle by which the steering is turned.
- When the vehicle is not in automatic mode, the steering gives continuous data about the steering angle to the control module for other functional modules to use

Diagnostic Information Generation:

- Controller failure will capture a blackbox and all the request messages happened from the stack trace will be stored as a hardware (SSD) inside a physical black box.
- This SSD can be detected and used for diagnostic information.
- Monitoring mobile application will also receive a stack of messages from the controller when there is a failure or periodic maintenance required.

Non Functional - Security





- No component can see or manipulate the other components.
- There will be one configuration module that defines the workflow, and in that file, we will mention all the security attributes.
- Security attributes consist of what neighboring components can communicate with the specific part and to what scope.
- The configuration file will be encrypted and can be deployed along with the application.
- The unencrypted configuration will be confidential and maintained along with the other confidential information.

Non Functional - Portability

- The System is designed in such a way that each module is an independent microservice that can be controlled by Application Programming Interface (API)
- So Each Part of the System can be decoupled and used individually
- There will be one controller module that makes use of all the other modules that make up the application
- In this way, we could update each module separately with the version that could support backward compatibility.
- There will be one configuration file that defines the workflow of the application

Non Functional - Reliability

The system has been designed to be reliable and sturdy.

Reliability is the probability that a system will perform satisfactorily for at least a given period of time when used under stated conditions.

It will have a Mean Time Between Failures of not less than 12 months.This can be achieved by having parallel and series structures of indicators.

In many cases, a single station is insufficient for navigation. Hence the navigation receivers are characterized by two MTBF values and by two common installation configurations, single and dual. The reliability of navigation receivers varies with type of system unit model, manufacturer, type of use, and age.

Determination of MTBF values for all receivers of interest is tested and configured.

Non Functional - Availability

The system is designed to be available even during the period when there's no internet connection for the gps system.

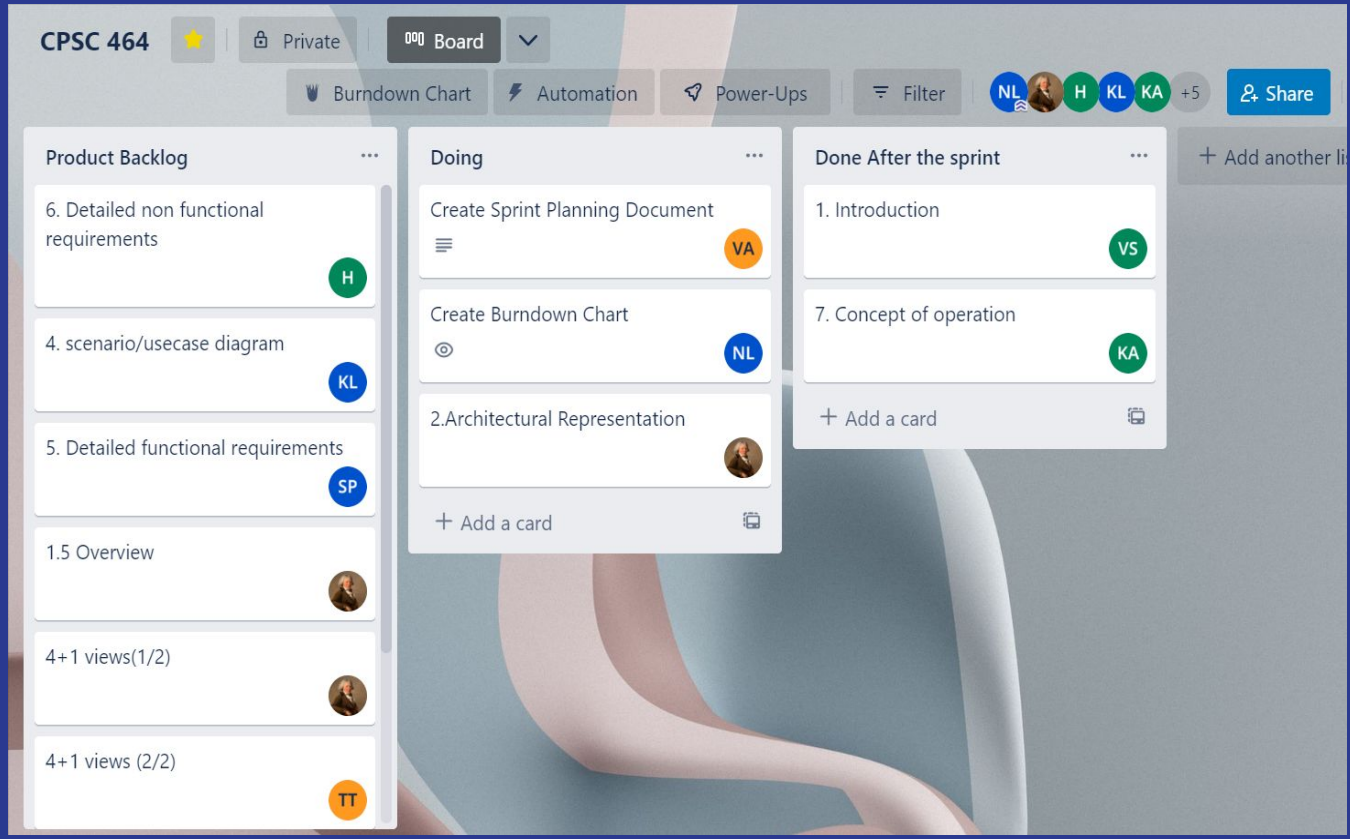
Even during offline mode the system has been enabled to work efficiently with storage in the local database system

The percentage of of time over any 24 hours interval that the predicted positioning from the satellites is less than 95% given at any point of the time

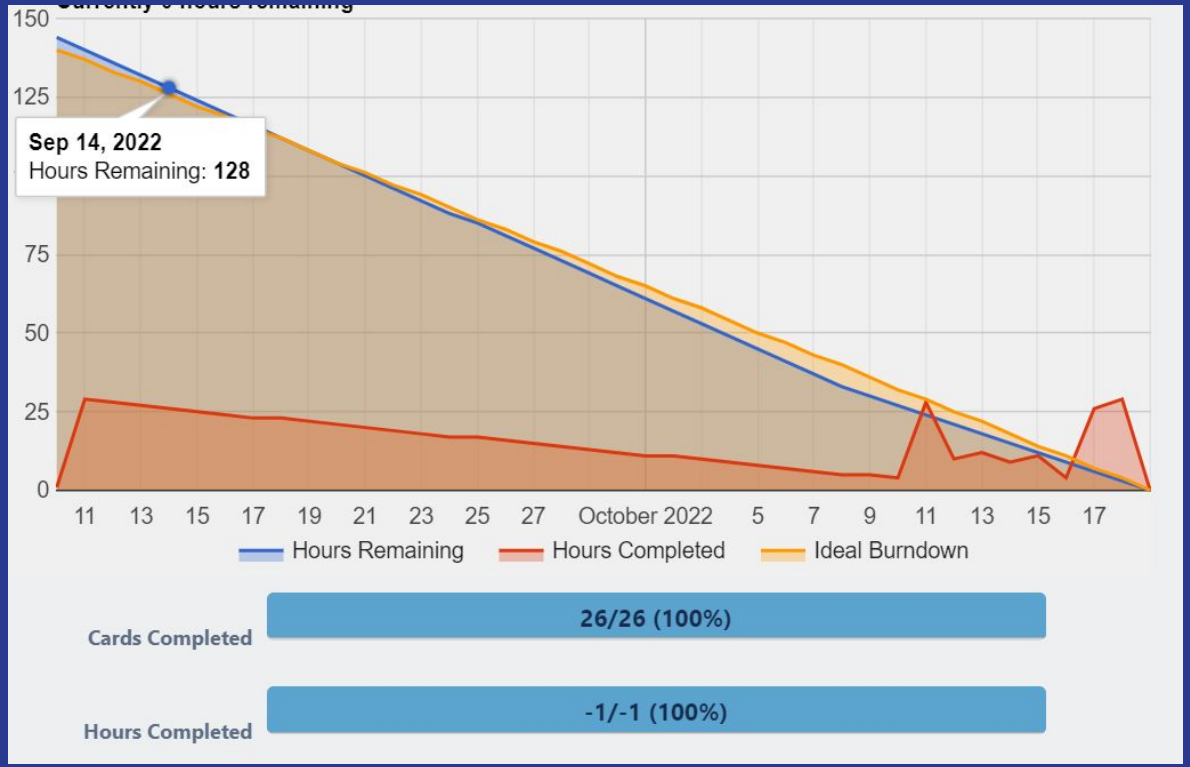
The system has been designed to provide the Navigation Display Update Messages by testing with different scenarios and obstacle testing to provide right 99.9999 of the scenarios

Titan Scrum Artifacts

Sprint Backlog



Burndown Charts

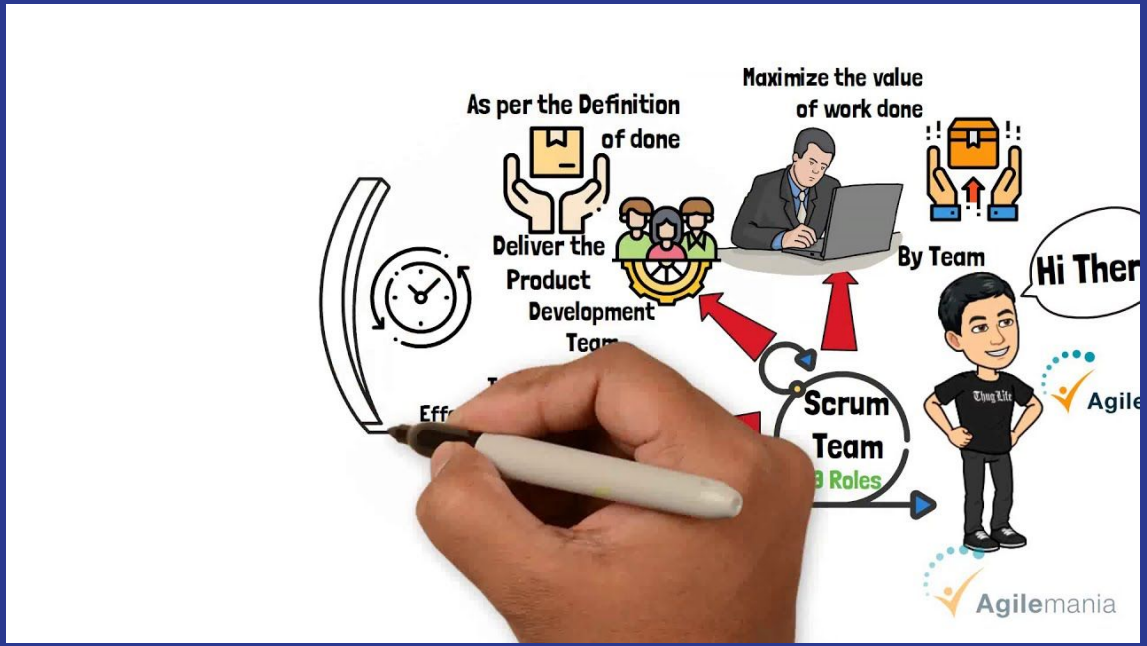


Titan’s Scrum Team

1. Annamalai Arumugam (PO)
2. Nitish (SM)

Active Agile Scrum Team Members

3. Kishore
4. Sneha
5. Hetal
6. Ben
7. Thomas
8. Vinay
9. Keshav



Questions

References

Images :

- https://i.ytimg.com/vi/WjWPsVdRj0A/maxresdefault.jpg
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRIlQdEUuA1SpbNkJxe1Kl4A8uKSSBk0QE9sw&usqp=CAU
- https://go.isostech.com/hubfs/Untitled-38.png

Thank you for your patience :)