# 

**Software Engineering Assignment**

**Traffic Control System**

**Module Title: Software Engineering (B9IS100)**

**Module Leader: Mr. Harnaik Dhoot**

**Group Members: Manik Mahashabde (10518579)**

**Saurabh Devade (10531140)**

**Sabitha Maram (10533048)**

Table of Contents

[1. Introduction 2](#_Toc26707332)

[2. Choice Of Process Model 3](#_Toc26707333)

[2.1 About Scrum 3](#_Toc26707334)

[2.2 Why We Used Scrum? 5](#_Toc26707335)

[3. Development of TCS using Scrum 5](#_Toc26707336)

[3.1 Project planning, analysis, and assumptions](#_Toc26707337)

[3.2 Release planning 6](#_Toc26707338)

[4. Sprint Cycles 7](#_Toc26707339)

[4.1 Sprint 1 [2 Weeks] 7](#_Toc26707340)

[4.2 Sprint 2 [2 Weeks] 8](#_Toc26707341)

[4.3 Sprint 3 [2 weeks] 10](#_Toc26707342)

[4.4 Sprint 4 [2 weeks] 11](#_Toc26707343)

[4.5 Sprint 5 [2 weeks] 12](#_Toc26707344)

[4.6 Sprint 6 [2 weeks] 13](#_Toc26707345)

[5. Architecture Design Diagram 14](#_Toc26707346)

[6. Type Of Architecture And Design Pattern 15](#_Toc26707347)

[7. UML Diagrams 16](#_Toc26707348)

[7.1 Class Diagram 16](#_Toc26707349)

[7.2 Activity Diagram 16](#_Toc26707350)

[7.3 Use Case Diagram 17](#_Toc26707351)

[7.4 Sequence Diagram 18](#_Toc26707352)

[8. Test Cases 20](#_Toc26707353)

[9. Risk Plan 22](#_Toc26707354)

[9.1 Risk 1: 22](#_Toc26707355)

[9.2 Risk 2: 23](#_Toc26707356)

[9.3 Risk 3: 23](#_Toc26707357)

[9.4 Risk 4 23](#_Toc26707358)

[References & Bibliography 24](#_Toc26707359)

[APPENDIX 25](#_Toc26707360)

[A.1 Code of signal status 25](#_Toc26707361)

# Table of Figures

[Figure 1 - The Scrum framework 4](#_Toc26712662)

[Figure 2 - Architecture of the application 14](#_Toc26712663)

[Figure 3 - Class Diagram 16](#_Toc26712664)

[Figure 4 - Activity Diagram 17](#_Toc26712665)

[Figure 5 - Use Case Diagram 18](#_Toc26712666)

[Figure 6 - Sequence Diagram 19](#_Toc26712667)

# List of Tables

Table 1 – Release Plan Table 6-7

# 1. Introduction

In this project, we have designed a smart traffic management system termed as **Traffic Control System**. We have designed this system for our officer **Dublin City Council(DCC)** so that they can manage the Dublin city traffic in a more efficient manner. It is specially designed for pedestrian crossing and giving the highest priority to emergency vehicles such as ambulance, police, fire brigade, security forces for crossing the signal. On a higher level on the road, the sensors we have used are **Acoustic** **detectors** at each 500m distance all over the road which receives the incoming sound frequencies of different vehicles and gives the information of sound frequency, time, etc. to the next sensor. At each signal, we have a parent sensor node which receives the information of nearby sensor and send the information to the system. After that congestion is calculated accordingly and action is taken to change the signal status of the pedestrian signal and vehicle signal. The signal state is changed after a certain amount of time depending upon the current congestion. If in case any emergency vehicle approaches the signal our sensors detect them through their high-frequency noise and pass the information further so that the parent node sensor can send the information to the system. Upon receiving emergency vehicle sensor data our system overrides the existing congestion calculation functionality and changes the signals. In this case, both pedestrian signal and vehicle signal status are changed to **RED** by the system.

We also have a camera installed on each signal. These cameras contain the functionality of image processing which can identify normal vehicles, pedestrians and emergency vehicles. All of this data is stored in the database which is stored in DCC secured data center. The database system is a MUMPS database. The database server is **Linux**. This data is available to DCC so that they can use later for legal actions.

For designing applications, we have a team, four developers. Two developers out of 4 are senior developers having 5 and 7 years of experience in IT, one developer is mid-level developer having experience of 3 years in IT and lastly, we have a junior developer who has 1 year of experience in IT. We have a manager who is having four years of experience developing software development projects out of which for 18 months he was a certified **Scrum** Master. We have used an **Agile** process methodology for developing our system. Work is given to each developer accordingly to their experience. If it is a difficult and high priority task it is given to senior developer and if it is an easy task then it is assigned to a junior developer.

# 2. Choice Of Process Model

The process model we have used is **Scrum** which is an Agile development methodology. Scrum methodology was first introduced by Takeuchi and Nonaka (1986) based on its successful usage in the manufacturing industries. Scrum process model encourages the team to do well in their tasks by tracking progress and giving specific timelines to achieve their targets. (Srivastava, Bhardwaj and Saraswat, 2017)

## 2.1 About Scrum

*There are various components of scrum such as:*

**Product backlog**: It is prepared at the starting of the project. It consists of all the requirements in the form of user stories which will be covered in the entire project. (Atlassian, “Product Backlog Grooming”,2019)

**Sprint backlog**: It is prepared at the start of each cycle or **Sprint** in terms of the scrum process model. Each user story is divided into tasks and added in the sprint backlog.

**Sprint**: It is a duration in which a set of user stories will be covered. Each developer will be assigned a task from a sprint backlog. The time of each sprint is usually 1-4 weeks. Entire project is divided and covered in multiple sprints (Atlassian, “Sprints | Atlassian”)

**Increment**: It refers to the amount of work completed at the end of each sprint where the team demonstrates what has been completed in the current sprint.

**Daily stand up:** It is a daily meeting of 10-15 mins kept by scrum master. In this meeting each developer or member updates their current task status. (Atlassian, “Agile Daily Standup”)

**Sprint retrospective**: It is meeting held at the end of each sprint where the team comes together and check what functionality is working and which ones are giving issues.

*Important people in a scrum are:*

**Product owner**: They are responsible for understanding the business or customer requirements clearly and explaining them to the development team. They also make the product backlog by adding user stories and also decided what part of the project should be covered in each release iteration. (Atlassian, “Agile Scrum Roles | Atlassian”)

**Scrum Master**: They are responsible to make sure that the entire team is working on track according to a sprint timeline. They arrange meetings such as stand-ups, sprint retrospective, sprint release, sprint planning. They also need to have in-depth knowledge of the tasks of each team member so that they can work with developers to make the solution more optimize. (Atlassian, “What Is a Scrum Master? | Atlassian”,2018)

**Scrum development team**: This team is responsible for doing all work and achieving the deliverables. Team consists usually of 4-8 people who have different skillset and experience. (Atlassian, “Agile Scrum Roles | Atlassian”)

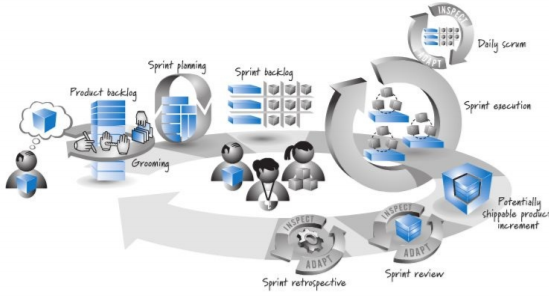


Figure 1 - The Scrum framework

## 2.2 Why We Used Scrum?

The reason for using the scrum process model is that it is widely used all over the world and also easy to get accustomed to a team (Nootyaskool and Ounsrimuang, 2016). As we are developing Traffic Control System we divided different functionalities to be covered in each iteration or sprint. As the length of each sprint in Scrum is 1-4 weeks and just 2 weeks in our case. Hence the sprint cycle is shorter as compared to another agile process, for example, **Extreme Programming**. As a result, a team of 4 has a good track of short targets within a sprint. Each story is divided into tasks and each task is assigned to the team member according to their experience and difficulty level of the task. Once we achieved partial functionality of the TCS system we gave the first release and completed the project in various iterations. We kept our project manager as Scrum Master and a member from DCC as a product owner. This process model gave a good understanding of the entire TCS process to our team and it also made sure that the entire team of 4 is on track and motivated to complete their individual tasks. Using Scrum process model scrum master also made sure that none of the people in the team felt overburdened and is given a task of difficulty level as per his/her capabilities and it is also completed on time.

# 3. Development of TCS using Scrum

## 3.1 Project planning, analysis, and assumptions

Now we are going to start the implementation of TCS. The first step is to create a product backlog of user stories.

After gathering requirements from DCC the team came up with the following user stories. ***Product backlog*** *of user stories is shown below:*

1. As a DCC officer, I want the TCS system to send the data (fetched at parent node sensor) to the database in the DCC secured datacenter so that it can be inserted in the database. **(functional requirement)**
2. As a DCC officer, I want the TCS system to send the video data (fetched from the video cameras )to the DCC secured database so that it can be used for legal reasons **(functional requirement)**
3. As a DCC officer, I want the TCS system to insert video, sensor, and EVAS data in the DCC secured MUMPS database so that it can be retrieved and updated later **(functional requirement)**
4. As a DCC officer, I want to retrieve data from the DCC secured database so that I can keep track of legal reasons**(functional requirement)**.
5. As a DCC officer, I want the TCS system to be able to increase the performance of sensor data conversion and transfer to the database so that operation speed is increased. **(non-functional requirement)**
6. As a DCC officer, I want the TCS system to be able to increase the performance of video data transfer so that video processing speed can be increased **(non-functional requirement)**
7. As a DCC officer, I want the TCS system to interact with signals based on sensor data so that signals can be changed **(functional requirement)**
8. As a pedestrian, I want to press the signal so that I can cross the road based on traffic congestion **(functional requirement)**
9. As an emergency vehicle driver, I want the EVAS system to override the existing sensor functionality so that I can easily cross the road. **(functional requirement)**
10. As a DCC officer, I want the TCS system to add EVAS sensor data into the database in the DCC data center so that it can be used by DCC **(functional requirement)**
11. As a DCC officer, I want the TCS system to have default functionality for my pedestrian signals and vehicle signals so that normal operations at the signals can be performed **(functional requirement)**
12. As a DCC officer, I want the TCS system to have fast insertion and retrievals time in the MUMPS database so that the database system is efficient. **(non-functional requirement)**

Project duration is going to be of **3 months** where each sprint is for **2 weeks**. We are going to cover 2 user stories in each sprint. Our **first release** will be coming at the end of 2 months or after 3 sprints. Now let’s divide user stories in each sprint cycle. We have kept the **assumptions** that the sensors, cameras are going to use internet connectivity provided by the **Government of Ireland** to communicate with the TCS system and database. Each signal is having a **TCS system** that acts as the main processing unit for the sensor, videos, and EVAS data. It controls the functionality of receiving input from the parent node sensor, image processed data from camera and EVAS transmitter signal given by EVAS drivers. TCS system is also responsible for changing the pedestrian signal and vehicle signal accordingly. Then the TCS system also sends the data to the database tables in the DCC data center where it is accessed by a DCC client.

## 3.2 Release planning

Release planning for the entire project is divided into four releases: first release, the second release, third release, and final release.

*Release Goals:*

1. **First release** after parent node sensor and video data can be recorded in the database and pedestrian signal and vehicle signal both change every minute according to the default time set.
2. **Second release** after TCS signals can interact with sensor data and change the signals.
3. **Third release** after pedestrian and EVAS can communicate with TCS.
4. **Final release** after all the stories are done.

*Below is the To-do list as per release plan:*

|  |  |  |
| --- | --- | --- |
| **User story ID** | **User story** | **Priority** |
| 3 | As a DCC officer, I want the TCS system to insert video, sensor, and EVAS data in the DCC secured MUMPS database so that it can be retrieved and updated later | 1 |
| 12 | As a DCC officer, I want the TCS system to have fast insertion and retrievals time in the MUMPS database so that the database system is efficient. | 2 |
| 11 | As a DCC officer, I want the TCS system to have default functionality for my pedestrian signals and vehicle signals so that normal operations at the signals can be performed. | 3 |
| 1 | As a DCC officer, I want the TCS system to send the data (fetched at parent node sensor) to the database in the DCC secured datacenter so that it can be inserted in the database. | 4 |
| 5 | As a DCC officer, I want the TCS system to be able to increase the performance of sensor data conversion and transfer to the database so that operation speed is increased. | 5 |
| 2 | As a DCC officer, I want the TCS system to send the video data (fetched from the video cameras )to the DCC secured database so that it can be used for legal reasons | 6 |
| **----------------** | **----------------FIRST RELEASE------------** | **-----------------** |
| 6 | As a DCC officer, I want the TCS system to be able to increase the performance of video data transfer so that video processing speed can be increased. | 7 |
| 7 | As a DCC officer, I want the TCS system to interact with signals based on sensor data so that signals can be changed | 8 |
| **----------------** | **----------------SECOND RELEASE---------------** | **-----------------** |
| 8 | As a pedestrian, I want to press the signal so that I can cross the road based on traffic congestion. | 9 |
| 9 | As an emergency vehicle driver, I want the EVAS system to override the existing sensor functionality so that I can easily cross the road | 10 |
| **----------------** | **----------------THIRD RELEASE---------------** | **-----------------** |
| 10 | As a DCC officer, I want the TCS system to add EVAS sensor data into the database in the DCC data center so that it can be used by DCC | 11 |
| 4 | As a DCC officer, I want to retrieve data from the DCC secured database so that I can keep a track | 12 |
| **----------------** | **----------------FINAL RELEASE---------------** | **-----------------** |

*Table 1: Release plan table*

*First Release:*

As can be seen from the above table first release will be after the **3rd sprint** and after completion of user stories **3,12,11,1,5 and 2**.

*Second Release:*

The second release will be after 4th sprint and after completion of user stories 6 and 7.

*Third Release:*

The third release will be after the 5th sprint and after completion of user stories 8 and 9.

*Final Release:*

The third release will be after the 6th sprint and after completion of user stories 10 and 4.

# 4. Sprint Cycles

Below are the 6 sprint cycles over the duration of the entire project.

## 4.1 Sprint 1: [2 Weeks]

In the first sprint, we are going to cover **story no. 3** and **story no. 12** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story*:** As a DCC officer, I want the TCS system to insert video, sensor, and EVAS data in the DCC secured MUMPS database so that it can be retrieved and updated later.

**Task 1**- Write a SQL query to create 3 tables **sensorData, videoData, EVASdata** in the database **TCS\_db.**

This task of table creation is done **only once** by a junior developer.

**Task 2**- Write a Java class containing function **receiveSensorData()** which receives incoming data from the TCS system at signal and converts it into SQL query using stored procedures for inserting into **sensorData** table.

This task is done by 1st senior developer. The program and query will execute every time it receives incoming data.

**Task 3**- Write a Java class containing function **receiveVideoData()** which receives incoming data from the TCS system at signal and converts it into SQL query using stored procedures for inserting into **videoData** table. The table stores the data and creates a new row for every **5 minutes** of video data. Hence **12 files per hour**.

This task is done by the 2nd senior developer. The program and query will execute every time it receives incoming data.

**Task 4**- Write a Java class containing function **receiveEVASData()** which receives incoming data from the TCS system (if it is an EVAS vehicle) at signal and converts it into SQL query using stored procedures for inserting into **EVASdata** table.

This task is done by a mid-level developer. The program and query will execute every time it receives incoming data.

***User Story***: As a DCC officer, I want the TCS system to have fast insertion and retrievals time in the MUMPS database so that the database system is efficient.

**Task 1**- Optimizing transfer time for inserting sensor, video, EVAS data into **sensorData, videoData, EVASdata** tables in the database **TCS\_db**

This task is done by 1st senior developer by testing various sample data to reduce time and increase efficiency.

**Task 2**- Optimizing transfer time for retrieving sensor, video, EVAS data from **sensorData, videoData, EVASdata** tables in the database **TCS\_db**

This task is done by 2nd senior developer by testing various sample data to reduce time and increase efficiency.

## 4.2 Sprint 2 : [2 Weeks]

In the 2nd sprint, we are going to cover **story no. 11** and **story no. 1** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story***: As a DCC officer, I want the TCS system to have default functionality for my pedestrian signals and vehicle signals so that normal operations at the signals can be performed

**Task 1**- Write a program to set the default change for vehicle signal after a specific time.

In this task we are going to create a Java class containing program **defaultVehicleSignal()** which will change the signal from **RED->AMBER->GREEN** or **GREEN->AMBER->RED** every minute. It will be GREEN for 1 minute and RED for 30 seconds. This task will be achieved by the 1st senior developer.

**Task 2**- Write a function to set the default change for a pedestrian signal after a specific time.

In this task, we are going to create a Java class containing function **defaultPedestrianSignal()** which will change the signal from **GREEN->AMBER->RED** or **RED->AMBER->GREEN** every minute. It will be RED for 1 minute and GREEN for 30 seconds. The vehicle signal state will be opposite to that of the pedestrian signal state except for **emergency** vehicles when both the signals will be **RED**. This task will be achieved by the 2nd senior developer.

**Task 3**- UnitTesting of **defaultVehicleSignal()** functionality.

It will be done by a mid-level developer.

**Task 4**- UnitTesting of **defaultPedestrianSignal ()** functionality.

It will be done by a junior developer.

***User Story*:** As a DCC officer, I want the TCS system to send the data (fetched at parent node sensor) to the database in the DCC secured datacenter so that it can be inserted in the database.

**Task 1**- Write a Java class containing function **sensorToTCS()** to send the data collected from the parent node sensor to TCS at signal. This task is done by 1st senior developer.

Once the parent node sensor receives data from the nearest acoustic detector sensor. It sends it to the TCS system.

**Task 2**- Write a Java class containing function **analogToDigital()** to change analog data received from the parent node sensor at TCS to digital data.

Once the TCS system received data from the parent node sensor. The data is in the form of analog as sensors receives analog data. Our program converts the analog data to digital data. So that the digital data will be stored in the **sensorData** table in the database. This task is done by 1st senior developer.

**Task 3**- Write a Java class containing function **generateSensorKey()** for generating a unique key **sensor-key** for the sensor record by appending sensor\_id and with the current timestamp.

We have assumed that each parent node sensor at the signal will have a unique **sensor\_id.** The program will generate a unique key by using sensor\_id with the current timestamp. We have generated the key because data in the MUMPS database is stored in the form of a key-value pair. This task is done by the 2nd senior developer.

**Task 4**- Develop a Java class containing function **sendTosensorData()** to hit the API at the DCC database.

Data from TCS System is sent in the form of JSON( JavaScript object notation). It is in key-value pair. The value here is the digital signal. This data will be received by our function **receiveSensorData().** This task is done by a mid-level developer.

**Task 5** - UnitTesting of our functions **analogToDigital(), generateSensorKey()** and **sendTosensorData().**

This task is done by a junior developer.

## 4.3 Sprint 3 [2 weeks]

In the 3rd sprint, we are going to cover **story no. 5** and **story no. 2** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story***: As a DCC officer, I want the TCS system to be able to increase the performance of sensor data conversion and transfer to the database so that operation speed is increased.

**Task 1**- Optimizing execution time for our functions **analogToDigital()**

This task is done by 1st senior developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

**Task 2**- Optimizing execution time for our functions **generateSensorKey()**

This task is done by 2nd senior developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

**Task 3**- Optimizing execution time for our functions **sendTosensorData()**

This task is done by the mid-level developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

***User Story***: As a DCC officer, I want the TCS system to send the video data (fetched from the video cameras )to the DCC secured database so that it can be used for legal reasons.

In this user story, we are using our camera to make real-time decisions based on the image processed output from our cameras. We are assuming that we have a camera that is capable of identifying pedestrians, normal vehicles and emergency vehicles using image processing through the **OpenCV** library. As an output TCS system is manipulating signals change timings in real-time. Each signal is having a camera with a unique id

**Task 1**- Write a Java class containing function **videoToTCS()** for generating a unique key for the camera record by appending **camera\_id** and with the current timestamp.

The video camera at the signal sends image processed data to the TCS system so that it can be used further. This task is done by 1st senior developer.

**Task 2**- Write a Java class containing function **generateVideoKey()** for generating a unique key for the camera record by appending **camera\_id** and with the current timestamp for every 10 minutes.

In this task, the image processed recording is used as records. Unique key **video-key** is generated by using camera id and current timestamp. This task is done by the 2nd senior developer.

**Task 3**- Write a Java class containing function **changeExistingSignal()** for changing the status of signal based on real-time.

In this task signal change, real-time decisions are taken by TCS. If in case the camera detects more pedestrians or different in count of vehicles as detects by the sensor. TCS system will manipulate the existing signal status defined by the parent node sensor. It will be 0 if signals are not changed initially. It will be more understandable in sprint 5.

**Task 4**- Develop a Java class containing function **sendTovideoData()** to hit the API at the DCC database.

The value here is the image processed video created using **Java base64 encode**. This data will be received by our function **receiveVideoData().** Thisfunction is called at the end of every **5 minutes** so that 5minutes of video data can be sent once it is collected at the TCS system. This task is done by a mid-level developer.

**Task 5-** UnitTesting of functions **generateVideoKey(), changeExistingSignal()** and **sendTovideoData()**

It will be done by junior developer.

## 4.4 Sprint 4 [2 weeks]

In the 4th sprint, we are going to cover **story no. 6** and **story no. 7** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story***: As a DCC officer, I want the TCS system to be able to increase the performance of video data transfer so that video processing speed can be increased.

**Task 1**- Optimizing execution time for our functions **generateVideoKey()**

This task is done by 1st senior developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

**Task 2**- Optimizing execution time for our functions **changeExistingSignal()**

This task is done by 2nd senior developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

**Task 3**- Optimizing execution time for our functions **sendTovideoData()**

This task is done by the mid-level developer by tracking the time of the function and making suitable changes in the logic to reduce time complexity **O.**

***User Story***: As a DCC officer, I want the TCS system to interact with signals based on sensor data so that signals can be changed

**Task 1**- Developing a Java class containing program **countCarRecords()** to calculate the extent of congestion based on parent sensor node data.

In this task whatever inputs received at the TCS system from the parent node sensor (which it receives from the nearest sensor), it calculates the total count of input cars and estimates congestion. This task is done by 1st senior developer.

**Task 2**- Developing a Java class containing program **signal change time()** to calculate optimal signal change time.

In this task, optimal signal change time is calculated based on the output of function **countCarRecords().** The time is given based on the extent of congestion :

Scenario **1**: If countCarRecords are more than 20 then the **pedestrian signal** will be RED->GREEN after **30 seconds** with respect to the current time and **vehicle signal** will be GREEN->RED.

Scenario **2**: If countCarRecords are more than 10 and less than 20 then the **pedestrian signal** will be RED-> GREEN after **20 seconds** with respect to the current time and **vehicle signal** will be GREEN->RED.

Scenario **3**: If countCarRecords are less than 10 then the pedestrian signal will be RED-> GREEN after **10 seconds** with respect to the current time and vehicle signal will be GREEN->RED.

Through this function, the TCS system knows when to change the state of signal and for how long.

This task is done by the 2nd senior developer.

**Task 3-** Writing a Java class containing program **changeSignal()** to change the pedestrian signal and vehicle signal.

In this program both the signals are changed after the time computed by function **signalChangeTime().**

This task is done by a mid-level developer. Now our function **changeExistingSignal()** in sprint 3 will also come into action. If in the case in real-time, there is any change in data of parent node signal at TCS system i.e. more pedestrian or vehicles are coming than expected. Then the function **changeExistingSignal()** will change the timings of pedestrian signal and vehicle signal accordingly.

**Task 4-** UnitTesting of functions **countCarRecords(), signalChangeTime()** and **changeSignal()**

It will be done by junior developer.

## 4.5 Sprint 5 [2 weeks]

In the 5th sprint we are going to cover **story no. 8** and **story no. 9** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story***: As a pedestrian, I want to press the signal so that I can cross the road based on traffic congestion.

**Task 1**- Developing a Java class containing program **registerInput()** to register user input from pedestrians.

In this task, the pedestrian will press the button and function **registerInput()** will be triggered at TCS system which will call the function **signalChangeTime()** internallyand both the signal will be changed according to output. Again, if suddenly more pedestrian appears then the camera will detect them and function **changeExistingSignal()** will change the existing signal timings. This task is done by a senior developer.

**Task 2-** Developing a Java class containing program **notifytoDCC().**

While there is a **RED** vehicle signal and vehicle tries to cross. TCS system will record it using the camera and buzzer will be given to DCC officers at UI portal.

This task will be done by mid-level developer

***User Story***: As an emergency vehicle driver, I want the EVAS system to override the existing sensor functionality so that I can easily cross the road.

**Task 1-**  Develop a Java class containing program **EVASdetect()** to detect the incoming emergency vehicle transmitter signal(given by driver) through the receiver present at the TCS system.

In case an emergency vehicle comes, a receiver at the TCS system at the signal will receive a message transmitted by the driver. The TCS system will then trigger a function **EVASdetect().** This task is done by a mid-level developer.

**Task 2-** Develop a Java class containing program **EVASsignalchange()** to override existing **changeSignal()** function.

After function **EVASdetect()** detects emergency vehicle **EVASsignalchange()** will override existing **changeSignal()** function and pedestrian signal and vehicle signal will be changed to **RED.**

**Task 2-** UnitTesting of functions **EVASdetect()** and **EVASsignalchange()**

This task will be done by a junior developer.

## 4.6 Sprint 6 [2 weeks]

In the last sprint we are going to cover **story no. 10** and **story no. 4** from the product backlog. The sprint backlog of the user stories and sprint planning is below:

***User Story***: As a DCC officer, I want the TCS system to add EVAS sensor data into the database in the DCC data center so that it can be used by DCC

**Task 1**- Develop a Java class containing function **sendToEVASData()** to hit the API at the DCC database.

Data is sent in the form of JSON( JavaScript object notation). It is in key-value pair. The value here is the digital signal of an emergency vehicle. This data will be received by the function **receiveEVASData()** mentioned in **sprint 1** which will add the data in table **EVASdata** in our database. This task is done by a mid-level developer.

**Task 2**- UnitTesting of function **sendToEVASData()**

This task will be done by a junior developer.

***User Story***: As a DCC officer, I want to retrieve data from the DCC secured database so that I can keep a track of it for legal reasons.

In this DCC can retrieve video, sensor, EVAS records according to the timestamp.

**Task 1**- Creating a Java class containing function **get video data()** for fetching video records and displaying to DCC.

In this, we are creating a function to get video data. A SQL select query is written which will be triggered internally and function will match the user input timestamp with the timestamp present in the key in table **videoData** and return the 5 minutes video data record to the user screen. This task is done by 1st senior developer.

**Task 2**- Creating a Java class containing function **getSensorData()** for fetching video records and displaying to DCC.

In this, we are creating a function to get sensor data. A SQL select query is written which will be triggered internally and function will match the user input timestamp with the timestamp present in the key in table **sensorData** and return the records on the user screen. This task is done by the 2nd senior developer.

**Task 3**- Creating a Java class containing function **getEVASData()** for fetching video records and displaying to DCC.

In this, we are creating a function to get EVAS data. A SQL select query is written which will be triggered internally and function will match the user input timestamp with the timestamp present in the key in table **EVASData** and return the records on the user screen. This task is done by a mid-level developer.

# 5. Architecture Design Diagram

It can be noted from sprint cycles that data collected by the TCS system from the parent code sensor, video cameras, and EVAS are continuously stored in respective tables in the database. It is available upon request from the DCC officers.

TCS system present at each signal receives sensor data from parent node sensor through function **sensorToTCS()**, video image processed data through function **videoToTCS()** and EVAS transmitter data through its receiver function **EVASdetect()**. It then change the pedestrian signal and vehicle signal accordingly by calling **changeSignal()** function. Simultaneously data is being sent to database **TCS\_db** via RestApi call in a JSON format. Functions for this are **sendToSensorData(), sendToVideoData()** and **sendToEVASData()** to the tables **sensorData, videoData** and **EVASdata** respectively in the form of key value pair. At the DCC end functions **receiveSensorData(), receiveVideoData()** and **receiveEVASData()**receiving incoming data from TCS and storing it using insert query.

Finally, in architecture, there is a user interface for DCC. They send a request for accessing sensor, video and EVAS data in the UI portal by using timestamp. Functions such as **getEVASData(), getVideoData()** and **getSensorData()** are triggered which matches the timestamp entered to that present in database and returns the data as JSON to the DCC.

Below is the high-level architecture diagram of the TCS application.

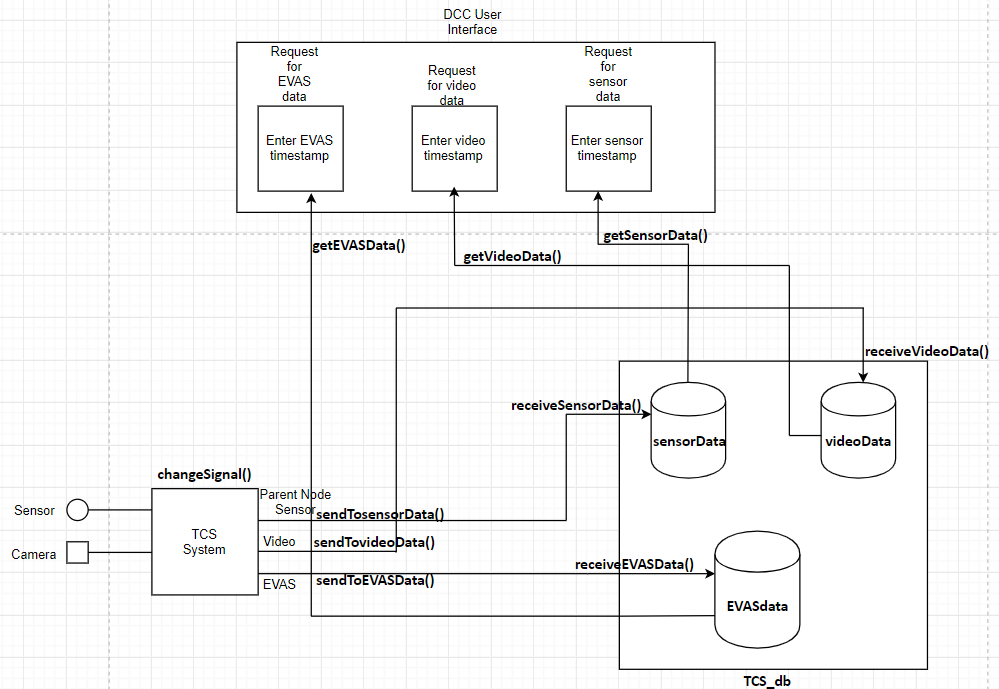


Figure - Architecture of the application

# 6. Type Of Architecture And Design Pattern

The architecture used here is **MVC** (Model-View-Controller) architecture. Details of its implementation are described below.

* **Model**:- TCS system situated of each signal is acting as a model. It receives the data from the parent node sensor, EVAS and camera present at the same signal. It hold the data and forwards it to the controller using classes containing functions **sendToSensorData()**, **sendToVideoData()** and **sendToEVASData().**
* **Controller** :-ClassFunctions present in DAO( data access object) such as **receiveSensorData(), receiveEVASData(),receiveVideoData(), getSensorData(), getEVASData()** and **getVideoData()** are acting as controller by managing the interaction between model and view. The controller is also responsible for storing and retrieving records in **TCS\_db** database. DAO provides a layer of abstraction on top of database for better accessibility of the tables.
* **View** :- The DCC user interface which consist of functionalities for accessing sensor, EVAS and video data from the database act as a view. Functions **getSensorData(), getEVASData()** and **getVideoData()** are responsible for it.

Reason for using MVC architecture (Socratic Solution, Medium, 12 July 2017):

1. **Asynchronous data transfer**: MVC architecture in our application supports asynchronous transfer techniques by continuously storing and retrieving data in the database. The system is not stopping intermediately in between as a result. All the functionalities such as storing data, retrieving data and changing signal and independent and do not affect each other.
2. **Ideal for large scale development**: As we need to design the TCS system for the entire Dublin. MVC architecture is an optimal approach to be used here. It provides a better way of giving a blueprint on which design can be implemented.
3. **Modification does not affect the entire system**: Another reason for using MVC is that if in case any new functionality is added in the system the entire architecture won’t be affected due to it.

**Singleton** design pattern is used for designing the TCS System. By the use of a singleton design pattern, only one instance of the class object is created every time (Soni, 2019). Hence, whenever we are creating objects of class for accessing the functions such as sendToSensorData(), sendToVideoData() , sendToEVASData(), receiveSensorData(), receiveEVASData(), receiveVideoData(), getSensorData(), getEVASData(). We make sure that only one instance of the class is created and the same instance is used again and again during function calls. Also, all the functions are **static** in nature and constructors are declared **private.** So that objects are not created using the **new** keyword. The use of this design pattern ensures that data is fully abstracted to the outside world. Hence the security of the system is intact. So, every time same instance of functions **sendToSensorData()**, **sendToVideoData()** and **sendToEVASData()** are used by TCS system for sending data to database in DCC data center. Each time key is created the same instance of sensor data function and video data functions are used which ensures abstraction to the outside world. Also, memory utilization is very minimal which reduces the space complexity and increases the performance of the system.

When our client DCC access the data using functions **getSensorData(), getEVASData()** and **getVideoData()** each time single instance is created to ensure usability and hiding the internal details. Use of singleton design pattern also risk of hacking as the constructor is declared as private hence new objects cannot be created. The only way to access methods is by using ***ClassName.MethodName().*** All the above-mentioned methods are **static** by default.

# 7. UML Diagrams

## 7.1 Class Diagram

In the UML class diagram, we have a TCS system represented as parent class. It has 3 subclasses Sensor, VideoData and EVAS. Each of the class has functions **changesignal(), changeExistingSignal(), EVASdetect()** which act as property of subclass. We have another signal class that has two subclasses Vehicle Signal and Pedestrian signal.

State of classes Sensor, VideoData and EVAS determines the type of signal in signals class. For the exam, if a sensor changes the pedestrian signal to **GREEN** then the vehicle signal state will be **RED.** If EVAS is detected then both the signals will be **RED**. Every TCS system is having a parent node sensor, camera, and EVAS. Hence there is a one to many mapping between TCS, sensor, and videoData.

*Class diagram is shown below:*

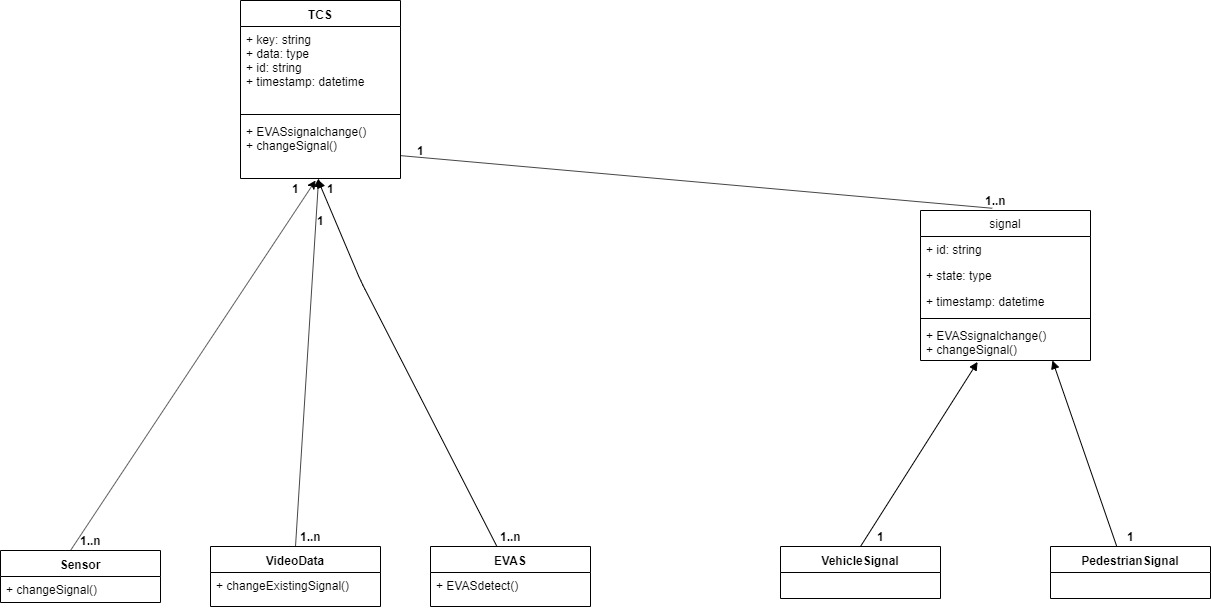


Figure - Class Diagram

## 7.2 Activity Diagram

In the initial state, it can be seen that pedestrian input is collected and reading of sensor data, video data is done at TCS. After that, it is congestion is calculated based on sensor and video data collected. Depending upon the congestion TCS system takes the decision when to change pedestrian signal to GREEN and vehicle to RED. If a case decision to change to RED is given to the vehicle signal then the pedestrian signal is changed to GREEN. Depending upon the extent of congestion pedestrian signal is changed after 30 seconds if the congestion count is high. It is changed after 20 sec if the congestion count is average and it is changed after 10 seconds if the congestion count is minimum. If it is an EVAS vehicle, the decision is taken to override the normal functionality to calculate congestion and both pedestrian signal and vehicle signal are changed to RED. Simultaneously action is taken to store all the three states i.e. signal data, video data, and EVAS data into the database. The final state of the system is reached upon access to data by DCC clients. Although it is a continuous process which doesn’t have any interruptions in between. Hence all the functionalities are happening simultaneously.

The activity diagram describes the system’s dynamic behavior by describing actions as events in the diagram starting from its initial state to all the internal actions or behavior and finally the end state of the system.

*Activity diagram is shown below:*

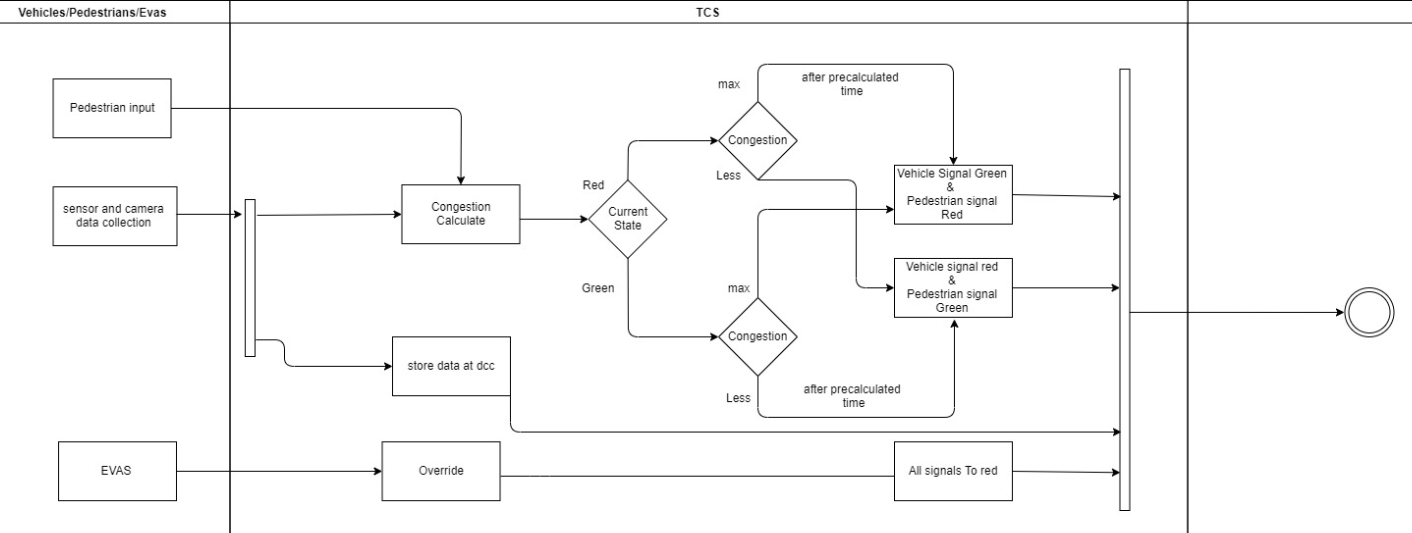


Figure - Activity Diagram

## 7.3 Use Case Diagram

The use case diagram is basically showing how user or actor will interact with the system. It also shows the involvement of the actor with different use cases in the system.

From the diagram shown below, it can be seen that when pedestrians and vehicles input the data to the system, it first calculates congestion. Then, after precalculated time system will display the specific signal based on its current signal state of pedestrian signals and vehicle signal

When EVAS arrives, all signals are by default turned to red

Also, all sensor data is stored at the DCC center.

*Use Case diagram is shown below:*

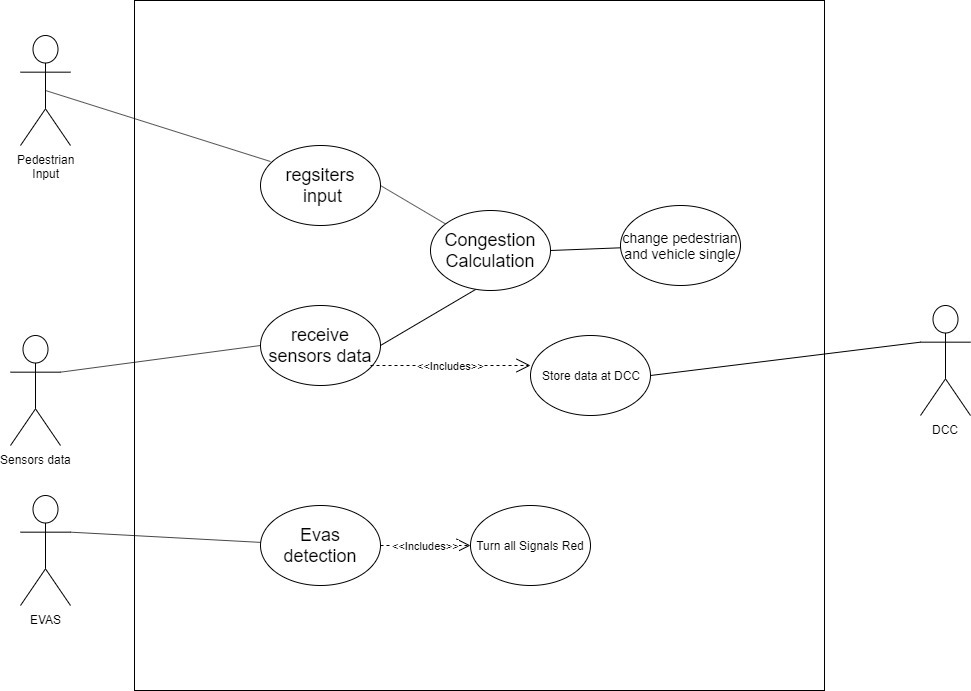


Figure - Use Case Diagram

## 7.4 Sequence Diagram

The sequence diagram as the name suggests signifies a sequence of events that takes place one after the other. Initially sensor data is collected at TCS using function **sensorToTCS().** Once that is done if it is data collected from EVAS both pedestrian signal and vehicle signals are changed to RED. Otherwise in case of any other vehicle **calculate congestion()** function is initiated which instantiates **countCarRecords()** function.

The time is given based on the extent of congestion:

Scenario **1**: If countCarRecords are more than 20 then the **pedestrian signal** will be RED->GREEN after **30 seconds** with respect to the current time and **vehicle signal** will be GREEN->RED.

Scenario **2**: If countCarRecords are more than 10 and less than 20 then the **pedestrian signal** will be RED-> GREEN after **20 seconds** with respect to the current time and **vehicle signal** will be GREEN->RED.

Scenario **3**: If countCarRecords are less than 10 then the pedestrian signal will be RED-> GREEN after **10 seconds** with respect to the current time and vehicle signal will be GREEN->RED.

Data is also stored at DCC as shown below:

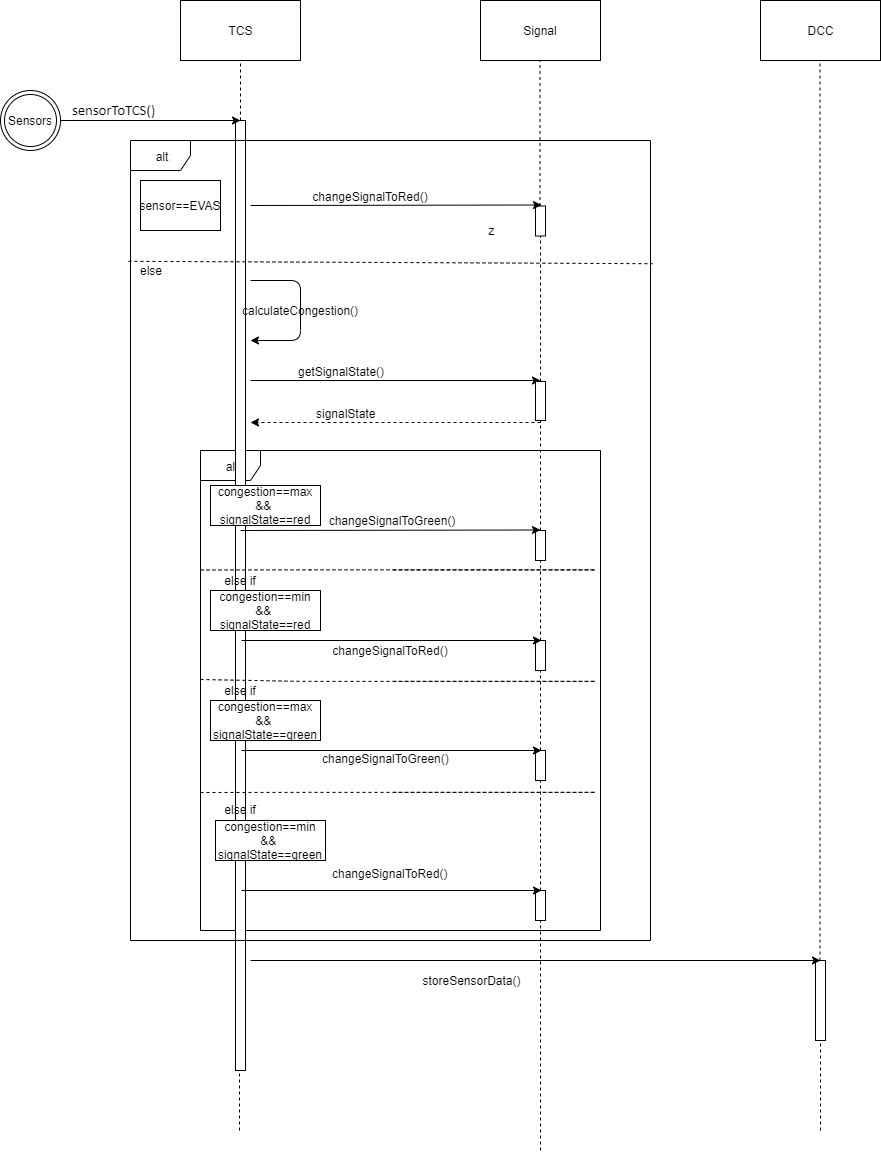


Figure - Sequence Diagram

# 8. Test Cases

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 1 | | **Test Case ID** | | | 1 | |
| **Test Case Description** | | EVAS Detection | | **Test Priority** | | | HIGH | |
| **Pre-Requisite** | | Working signals | | **Post-Requisite** | | | NA | |
| Test Execution Steps: | | | | | | | | |
| **Sr.No** | **Action** | | **Expected Output** | | **Actual Output** | **Test Result** | | **Test Comments** |
| 1 | Emergency vehicle arrives | | Vehicle signal should turn to red | | Vehicle signal turned to red | Success | | As soon as the EVAS arrived the signal tuned to red irrespective of current state |
| 2 | Emergency vehicle arrives | | Pedestrian signal should turn to red | | Pedestrian signal turned to red | Success | | When Evas arrived pedestrian signal turns to red |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 2 | | **Test Case ID** | | | 2 | |
| **Test Case Description** | | Pedestrian Signal working | | **Test Priority** | | | HIGH | |
| **Pre-Requisite** | | Working signals | | **Post-Requisite** | | | NA | |
| Test Execution Steps: | | | | | | | | |
| **Sr.No** | **Action** | | **Expected Output** | | **Actual Output** | **Test Result** | | **Test Comments** |
| 1 | Pedestrian presses the button | | If the pedestrian signal is red, it should turn to green after 1 minute | | After pressing the button, the pedestrian signal turned to green after 1 minute | Success | | As soon as the pedestrian presses the button, the signal turns to green in 1 minute. |
| 2 | Pedestrian signal turns to green | | After Pedestrian signal turns to green, after 30 seconds it should turn back to red | | Pedestrian signal turned to red after 30 seconds | Success | | Pedestrian signal turned to red after 30 seconds |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 3 | | **Test Case ID** | | | 3 | |
| **Test Case Description** | | Data retrieval for DCC officer | | **Test Priority** | | | HIGH | |
| **Pre-Requisite** | | Database containing data | | **Post-Requisite** | | | NA | |
| Test Execution Steps: | | | | | | | | |
| **Sr.No** | **Action** | | **Expected Output** | | **Actual Output** | **Test Result** | | **Test Comments** |
| 1 | DCC officer enters Timestamp | | If DCC officer enters timestamp for the particular record, the system should return the data associated with that timestamp | | Data returned successfully. | Success | | The test case worked. |
| 2 | DCC officer enters invalid Timestamp | | If DCC officer enters invalid timestamp, then the system should return the error message | | System crashed | Failure | | Instead of giving the error message, the system crashed |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 4 | | **Test Case ID** | | | 4 | |
| **Test Case Description** | | Data storage at DCC | | **Test Priority** | | | HIGH | |
| **Pre-Requisite** | | Working sensors, Internet Connection | | **Post-Requisite** | | | NA | |
| Test Execution Steps: | | | | | | | | |
| **Sr.No** | **Action** | | **Expected Output** | | **Actual Output** | **Test Result** | | **Test Comments** |
| 1 | Sensor sends the data | | Data stored at DCC Centre | | Data stored successfully. | Success | | The test case worked. |
| 2 | Video camera sends video, Low-speed internet | | Video stored at a database. | | Video storage failed because of low internet connection | Failure | | Error saving the video data |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 5 | | **Test Case ID** | | | 5 | |
| **Test Case Description** | | Calculation of time to change signal based on current traffic congestion | | **Test Priority** | | | HIGH | |
| **Pre-Requisite** | | Working sensors, Pedestrian presses button | | **Post-Requisite** | | | NA | |
| Test Execution Steps: | | | | | | | | |
| **Sr.No** | **Action** | | **Expected Output** | | **Actual Output** | **Test Result** | | **Test Comments** |
| 1 | Sensor detects that the number of vehicles is greater than 20 | | Pedestrian signal should turn to green after 30 seconds | | Pedestrian signal changed to green after 30 seconds | Success | | The test case worked. |
| 2 | Sensor detects that the number of vehicles is less than 10 | | Pedestrian signal should turn to green after 10 seconds | | Pedestrian signal changed to green after 10 seconds | Success | | This test has executed successfully |
| 3 | Sensor detects that the number of vehicles are more than 10 and less than 20 | | Pedestrian signal should turn to green after 20 seconds | | Pedestrian signal changed to green after 10 seconds | Failed | | Test case failed |

# 9. Risk Plan

Managing Risk is one of the most important aspects of every project and it should be given a valuable consideration. Risk is nothing but a situation you don’t want to occur while working on the project. In large applications, risk can be of any type like Organizational, Project risk, Product risk, etc. There are few steps to avoid this uncertainty while working on a project i.e. Risk Identification, Risk Analysis, Risk planning, and Risk Monitoring (Gonen, 2011).

We will have a look at a few risks that might occur in this project:

9.1 Risk 1: Cameras used for capturing and sensing objects stopped working properly.

**Type** - Technology

**Probability** – Moderate

**Severity** – RED (Serious)

It might happen that the cameras we are using for capturing live data is not working due to some technical difficulties, and because of that TCS will not be able to make decision for which signal to show depending on current traffic scenario. In this case, our system will fail completely.

**Plan** – In this case, there will be functionality implemented on the TCS system that, any of the component stops working, TCS will immediately notify the team with the signal id, that some technical difficulty has occurred on the signal. Once the team receives that notification, they will replace the defective camera with the new one.

## 9.2 Risk 2:

**Type** – The TCS motherboard on the signal is not working the way it should be.

**Probability** – Moderate

**Severity** – RED (Serious)

All the operation which are essentials in terms showing specific signals, and also sending live data to the DCC is performed on Motherboard which is installed on TCS. It is possible that during heavy weather conditions, some damage might happen to the motherboard and it will cause failure in the system or it will lose connection to the DCC.

**Plan** – To prevent this every Signal will be equipped with the secondary device which will perform only important operations until the defective unit is replaced with the new one. Also, if data which was not able to send, will be stored in onboard disk storage temporarily. And when the device replaced the data will be sent to the DCC.

9.3 Risk 3: The database being used for storing the sensor data and live videos every 5 minutes, is not appropriate in one or more aspects.

**Type** - Technology

**Probability** – Moderate

**Severity** – RED (Serious)

As mentioned in the business requirements we need to store the highly precise sensor data and live video into the database. By looking at the project it is clear that the system is huge because there are so many sensors and video cameras. So, the condition might occur that, the currently used database system is becoming inconvenient as it is not able to perform at high-speed rates. In this case, the system will collapse and there might be a loss of data.

**Plan** – In this scenario, the project manager will start looking for alternative high-performance database systems and he will approach the top-level management and will try to convince them to invest in the higher-performance database system.

9.4 Risk 4: One of the senior developers becomes sick just before delivering the sprint as per the schedule.

**Type** - People

**Probability** – Moderate

**Severity** – RED (Serious)

It is possible that during the project building stage, when the sprint delivering stage arrives, one of the senior developers falls sick and the project might get delay. It will increase the burden on the other team members and sprint delivery may get delayed.

**Plan** – To overcome this scenario, from the beginning both the senior developer will review each other’s work, so they will be well aware of each other’s job. So even if the one developer is not working the other can takes his work and if needed, they can arrange a phone call if others came across some difficulties.

Also, if this occurs project manager will reorganize the roles and distribute the work of unavailable developers among the others.

# *References & Bibliography*

Atlassian. “Agile Daily Standup.” Atlassian, 2019, www.atlassian.com/agile/scrum/standups. Accessed 7 Dec. 2019.

“Agile Scrum Roles | Atlassian.” Atlassian, 2019, www.atlassian.com/agile/scrum/roles. Accessed 8 Dec. 2019.

Gonen, A. (2011) ‘Optimal risk response plan of project risk management’, IEEE International Conference on Industrial Engineering and Engineering Management. IEEE, pp. 969–973. doi: 10.1109/IEEM.2011.6118060.

Nootyaskool, S. and Ounsrimuang, P. (2016) ‘Optimization algorithm using scrum process’, Proceedings - International Conference on Machine Learning and Cybernetics. IEEE, 1, pp. 245–250. doi: 10.1109/ICMLC.2016.7860908.

“Product Backlog Grooming.” Atlassian, 2019, www.atlassian.com/agile/scrum/backlogs.

“Sprints | Atlassian.” Atlassian, 2019, www.atlassian.com/agile/scrum/sprints.

Socratic Solution. “Why MVC Architecture?” Medium, Medium, 12 July 2017, medium.com/@socraticsol/why-mvc-architecture-e833e28e0c76. Accessed 8 Dec. 2019.

Soni, Devin. “What Is a Singleton?” Medium, Better Programming, 31 July 2019, medium.com/better-programming/what-is-a-singleton-2dc38ca08e92. Accessed 8 Dec. 2019.

Srivastava, A., Bhardwaj, S. and Saraswat, S. (2017) ‘SCRUM model for agile methodology’, Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017, 2017-Janua, pp. 864–869. doi: 10.1109/CCAA.2017.8229928.

“What Is a Scrum Master? | Atlassian.” Atlassian, 2018, www.atlassian.com/agile/scrum/scrum-master. Accessed 7 Dec. 2019.

# *APPENDIX*

A.1 Code of signal status

The code of signal status is available below the GitHub URL.

The main file which contains the code is TCS\_Signal\_Code.py. There are three input text files signal1.txt, signal2.txt, and signal3.txt. After processing these files according to given conditions, output is given in two files Output\_signal.txt and Error\_signal.txt.

https://github.com/Manik2018