**TRAFFIC CONTROL SYSTEM FOR DUBLIN CITY COUNCIL**

**DUBLIN BUSINESS SCHOOL**

**SOFTWARE ENGINEERING**

**AARON ANTHONY (10518734)**

**CLINTON DELVIN (10531003)**

**PADMARAJU G M (10522769)**

**FUNCTIONAL REQUIREMENTS:**

**Normal Colour changes**

Signal light must change from Red to Green. Green to Amber to Red.

**Highest priority for emergency vehicles (EVAS)**

As an Emergency Vehicle, it’s imperative that it reaches its destination on-time. Hence, we shall be assigning highest priority for signal changes to any Emergency Vehicles approaching.

**Traffic Control and Congestion with the help of Road Sensors and Cameras**

As a daily commuter, its necessary for traffic flow to be smooth during peak hours. Hence, we shall be using Cameras and Road sensors with the help pf Artificial Intelligence (AI) to alter signal timings and lighting appropriately.

**Pedestrian Crossing Signals**

As a pedestrian, waiting on a signal for long is frustrating and often leads to unwanted crossing of roads which in turn can lead to accidents. Hence, by assigning a high priority to pedestrian signals, we can ensure the quick change in pedestrian lights only when a pedestrian is present.

**Permission to override traffic system for critical scenarios.**

There are some exceptional cases where due to some unforeseen events like riots, earthquakes, parades etc certain roads or routes need to be blocked. Thus, our system can override these signals and the control will be passed onto DCC for further action. Although, priority for this is less than EVAS.

**Time-Based Traffic Control**

Traffic In the morning and evening hours is usually higher than normal hence, Roads leading towards the City and other workplaces, need to have appropriate Time-based Traffic Signals. At night the Traffic is more towards the suburbs and hence shall be adjusted appropriately.

**Switching to guided manual control.**

In case of an emergency, Garda/emergency services or any Authority (State level), must be able to take manual control of the signals to co-ordinate and manage the traffic flow. In such a case all control from DCC centre should be lost, to prevent cross talk.

**Alert system for signal violation.**

Drunk driving, Speed violation, Accidents, Signal violations or simply monitoring vehicles and pedestrians., all of these should be handled by the road sensors and cameras set up on these traffic signals. Where, the data is collected and sent to Dublin Central Traffic Control Database. The data shall be analysed and sent to further authorities for use by their respective departments.

**System revision through DCC only.**

Every city grows and expands rapidly. The number of roads, vehicles and people increase with time. Hence, we need to have a system which can handle revision of signal timings and colour changes along with addition/deletion of new signals.

**NON-FUNCTIONAL REQUIREMENTS**

**Priority for signals**

* Emergency vehicles with EVAS installed in them, should send a signal 1Km away from the approaching signals, so that the flow of traffic is handled efficiently.
* The override of signal control from DCC should be not more than 3 seconds. In the event of EVAS failing, the DCC should be notified by the EVAS system to perform an emergency override.
* When a pedestrian presses the crossing button, the signal across must be stopped within 1 minute to allow the pedestrian to cross. The pedestrian signal must be working for just one minute in this scenario.
* The priority of Pedestrian signal is below EVAS and Emergency calls.

**Road sensors & Camera storage and transfer**

* The Sensor arrays are available in 16K and 32K sizes. Hence, the distance covered by these will vary with the size of the arrays used. We shall use C/C++ to modulate and adjust the distance, quality and storage of these sensor arrays.
* The typical range should be around 50m. It should be able to detect presence of vehicles within range, measurement of volume, speed of vehicles and detect accidents.
* It should also support TCP/IP communication, LAN and WAN interface along with radio modems as well. So that data collected can be sent to the database for storage with a timestamp.
* The cameras installed must be HD 1080p along with Camera server capabilities such that the camera server can be directly linked to the DCC’s secure server.
* All the cameras must be GPS-time stamp enabled so that when vehicles pass by, it reads the license plate, timestamps it and sends it to the camera server which then encrypts the message and sends it over to the DCC’s secure server.
* It should also have capability to store data locally as well. GPRS linking between them ensures connectivity with each other through the city properly. And pull images only when required The Camera in-station enforcing speed must be at least 140mph so that all vehicles can be recorded with ease.
* It must also compare the current speed of the vehicle with the enforcement speed and raise a traffic violation ticket to DCC.
* Both devices need to be placed at an optimum height of 10m from the ground.

**Time-Based Traffic Control**

* Traffic signal’s optimum light changes must be set in advance depending upon the locality, time, density of vehicle, population and peculiar conditions like construction, poor roads etc.
* These pre-set default signal values are decided as an average of all the above factors. These default values can be modified by the DCC’s AI algorithm which shall run the dataset containing all the congestion reports from traffic signals and provide optimal light changes. These change in signal timings must be no more than 2 minutes. The change must return to default once the congestion goes below the threshold value.
* Any main road intersection depending on the number of roads:

1. **Four-way intersection:** Each traffic signal must be evenly timed preferably 3 minutes each along with pedestrian signal working for 1.5 Minutes in Synchronization with the adjacent signals.
2. **T-Junction:** The traffic signals which are on the main road must be evenly timed with longer pass time, that is, 4 minutes. Meanwhile, the T-intersection road entering must be half the time of the main road, that is 2 minutes. Pedestrian signals working for 1.5 minutes each in synchronization with adjacent signals.
3. **Traffic Circle:** These traffic signals will always be evenly timed to prevent any form of collision. The traffic signals will be made green in a clockwise manner so that every intersection can access the roundabout one at a time. Hence, each traffic signal will be on for 3 minutes each. Pedestrian signals shall work for 1.5 minutes anticlockwise.
4. **Y-intersection:** There shall be three traffic signals evenly timed at 3 minutes each. Pedestrian Signal at 1.5 minutes in synchronization with adjacent signals.
5. **Lanes & two-way roads:** Traffic signals shall have an evenly timed 2 minutes on-time before cutting to pedestrian signal of 1.5 minutes.

**PROCESS MODEL CHOSEN: SPIRAL**

The advantages of Spiral model over waterfall are as follows:

1. The spiral model is an evolutionary type of software life cycle model which merges the features of the prototype model and waterfall model. It has the potential for developing the incremental versions of the software by implementing the literateness of the prototype model and controlled methodical of the linear segmental model.
2. In Waterfall Model thorough understanding of requirements are essential from the beginning of the project which is difficult for complex or huge projects as the requirements can change and will have to be modified later depending on the scenario. Spiral model adapts to the change of requirements of the product. In our case, there are multiple requirements, modules, system requirements. Hence, development of a project as critical as this would become very rigid with waterfall model.
3. Waterfall model takes into consideration a limited amount of risks which are handled only at the test part. Hence, the risks coupled over the phases would be all compounded together with the final product making it very difficult to troubleshoot. The spiral model recognizes risk at earlier phases of the process which is revised or corrected at each iteration. Hence, for critical projects such as this, there are various risks which arise during the different phases of development and need to be addressed appropriately and immediately such that we move onto the next phase with minimal risk.
4. Waterfall model is suitable for small, simple and less complex systems as all the requirements have been specified or identified at an early stage, but when the system to be developed is complex, massive and critical in nature, we need to implement it phase wise, by initially prototyping, then implementing the design model, design validation, test plan, review, risk analysis, unit test and finally system test before releasing the final product.
5. A project becomes expensive in a waterfall model, when the final product is not as per the requirements and needs to be sent back to the drawing board. Which is avoided to a great extent in Spiral model.

**HOW WILL OUR TEAM ADHERE TO SPIRAL SYSTEM?**

1. Our first step will be to analyse the initial requirements and prototype the system to be developed on a very small-scale using development kits and basic circuitry along with some decent amount of code. The prototyping will ensure basic functioning of signals with respect to the timing criteria, obstruction identification changes, EVAS override and pedestrian priority switches. Once the prototyping is satisfactory enough, we shall start by analysing the requirements of the system. The various functional and non-functional requirements which decide the nature, capability, adaptability and constraint of the system. These requirements are mentioned above.
2. Next step is to decide the Development plan, since our company has 7 years of experience in Agile process development, we shall be using Influential Extreme Programming (XP) because of its key features, i.e. User stories for specification, refactoring, test-first development, pair programming, (Sommerville, 2014).
3. Now it is time to start implementing the system. We shall make a design model of the whole system which will be represented by UML diagrams in the following pages, which will outline the exact flow, working, dependencies, constraints and actors involved. Post this, we shall implement the design code for the traffic signals with the appropriate timing constraints, color changes, EVAS & pedestrian priorities, traffic flow monitoring and signal changes. The codes used to send and retrieve data from the database in M language.
4. Every design needs to be validated. Validation is concerned with demonstrating the consistency and completeness of design with respect to the user needs. This is the stage where we validate our system against our user requirements. According to Agile process development, user requirements is what it is all about and hence, the system needs to satisfy it.
5. Test plan. The most important phase of any system. Since, the system we are developing is extremely critical and cannot afford any kind of abrupt failure or other kinds of defects or bugs before it is put through to the manufacturing line. Our team shall be following the development testing. This ensures all the modules, objects, components work perfectly by itself, co-depending and along with each other. We shall also have interface testing since we have multiple interfaces with data flowing from one point to another. We shall finally use use-case testing which will ensure the system functions according to our system scenarios.
6. Until now, there should have been 2 sprints completed and we shall now review what all that has been done. Review happens after every sprint cycle. But a major review happens after every two sprint cycles. With the test plan decided. We shall perform risk analysis, which takes into consideration all the possible risks the system will face. Thus, suggesting new or better solutions to issues that may have developed over the phases.
7. Finally, Unit and System tests as mentioned above will be conducted and upon successful results, the system will be sent into production and implementation.

**TEST CASE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST-ID | TEST DESCRIPTION | TEST PROCEDURE | EXPECTED OUTPUT | TEST RESULT |
| 1 | Signal Transition  (With EVAS override equal to 0) | 1. Input file should read by the traffic controller. 2. Signal ID, EVAS override should be checked. 3. Colour from & Colour to with time should be read. | Colour must change as per the file instructions. | PASS |
| 2 | Signal Transition  (With EVAS override equal to 1) | 1. Input file should read by the traffic controller. 2. Signal ID, EVAS override should be checked. | Signal colour must turn to Red. | PASS |
| 3 | Live data collection from traffic controller | 1. Camera should send ping status to DCC with Camera ID. 2. DCC should select the Controller ID to view the Camera’s connected. 3. Select the Camera to view with Camera ID. | DCC should be able to view live feed of any Traffic controllers, Cameras. | PASS |
| 4 | Road Sensor detection and DCC communication  (Uplink) | 1. Sensor must be placed 10m above the ground. 2. Multiple obstructions up to 50m need to be placed in front of sensor. 3. Road sensor should be switched on. 4. Data should be sent to DCC with Road Sensor ID. 5. DCC should check if the road sensor reading is high. | The road sensor should detect vehicles up to 50m.  Data should be received and accessed by DCC. | PASS |
| 5 | DCC communication  (Downlink) and Traffic Controller | 1. Appropriate commands from DCC sent to the Traffic controller. 2. Traffic controller should read input file provided by the congestion algorithm. 3. Or Traffic controller will be under manual control by DCC. | The traffic controller should change the signal colours with time according to the algorithm files or manual overdrive. | PASS |
| 6 | Congestion Algorithm  Functioning | 1. Multiple values input into the software. 2. Algorithm run by DCC after authentication. 3. Input files generated to tackle congestion sent to traffic controllers. | Algorithm should change signal timings for more congested areas. | PASS |
| 7 | Manual Override | 1. Authentication by DCC head. 2. Select Controllers by ID or Area. 3. Provide any commands that is needed. | DCC should be able to control any traffic controller. | PASS |
| 8 | Signal Transition (Pedestrian) | 1. Pedestrian presses the button. | Pedestrian traffic light must turn green within 1 minute. | PASS |
| 9 | Database Connectivity | 1. Secure login through DCC official 2. Check for data present in Database. 3. Check if data can be viewed. 4. Check if data has been timestamped. | Database must be accessible only to DCC officials.  Videos, Congestion data should be accessed by DCC. | PASS |
| 10 | Traffic violation Ticketing | 1. Traffic violation data obtained from Traffic controller. 2. Violating vehicle data should be entered into the database. 3. Ticketing of the vehicles. | Violating vehicles will have a ticket raised against them. |  |

**RISK ANALYSIS**

In the spiral model that we are following risk analysis is a continuous process that happens every major sprint cycle. This way the risks are identified, understood and solved at every stage. The various risks predicted over the phases are:

1. Failure of electronic devices like Camera, Road sensors and Traffic light controllers. These electronic devices in their prototyping stages may encounter short-circuits, power surges, physical damages while prototyping them. Hence, to avoid these issues in the main system we will ensure that the devices will be secured in a hard-case shell. The wiring will be insulated twisted pair copper wires to prevent short-circuits and electro-magnetic effects. The memories of the devices will also be kept in check and once, data is sent from the device to the database it should be erased from the local memory.
2. Poor connectivity of data (GPRS, Internet) from the traffic controller to the DCC data centre. This will cause a delay in exchange of information, incorrect data transfer, slow manual override, etc. In short, Poor connectivity is not something ideal for a smart traffic control system. Hence, we must utilise extremely fast and reliable networking resources based on TCP/IP. Which ensures data being sent and received is correct.
3. In case of failure of an EVAS system in an emergency vehicle. The only method to keep the route clear, open or shut is by manual override. This can be done by manually by contacting DCC and getting permission for a manual override. This is a risk that emergency vehicles have in case EVAS system failure.
4. The MUMPS database should be backed onto the cloud to prevent any loss of data through physical damages. Since, MUMPS database has all its data globally accessible on any system and has the data stored on various computers.

1. Congestion Algorithm Failure is a major issue since, the traffic controller will behave in an erratic responsive mode, causing an absolute havoc and mess with the signal timings and colour changes, which in turn will cause a disaster with the city traffic. Hence, any type of congestion failure must be detected at an early stage and disabled, until the issue has been sorted.

**FIRST RELEASE PLAN**

Our company has been working with agile software for about 7 years. The team developing this project consists of 4 developers and a project manager who will also be the scrum master. Since, We have two senior developers with experience of 5 years and 7 years respectively. The entire team will be divided into three teams working under the scrum master. The development team will consist of the senior developer with 7 years experience working with aircraft control systems and the other senior developer with 5 years of experince. While, the testing team will consist of the mid-level developer with 3 years of experience and the junor developer with 1 year experience. The Scrum master will handle the Storyboard team. All three teams will work simultaneosuly to complete the project. The first release plan will be as follows:

1. The Scrum master gathers the user stories and system requirements with a lot of research, data, analysis and feedback over two sprint cycles. He may take assist from the Testing team here.
2. Half way through the first sprint cycle the development team will start prototyping the system as the user stories, requirements are gathered.
3. As the prototyping sprint is going on, the storyboard team gathers the system specifications and requirements needed for the main system.
4. Once the development team is successful with the prototyping, it will plan the development model and how it will go about implementing the project. The testing team also begins risk analysis of the prototype.
5. The development team begins development along with the testing team doing risk analysis and compononent testing simultaneoulsy after every sprint.
6. At the end of designing sprints, validation of system agaisnt the user stories is done by the testing team. The key feature of spiral model is the complete reliance on user stories. Hence, this is one of the most imprtant phases.
7. Test plan is decided immidiatley which will be reviewed in the the coming weeks.
8. Risk analysis is another major sprint cycle which is done by both the development and testing teams to search, solve and resolve any issus which arise along the dvelopment phase.
9. Testing team will conduct Unit tests and System tests over 2-3 sprint cycles.
10. The first release will be on October 28, 2020.

A screenshot of a cell phone

Description automatically generated

**UML DIAGRAMS**

**A close up of a map

Description automatically generatedFigure 1.**

* The Traffic Controller takes care of many responsibility like collecting the data, analyzing data, sending data, check what implementation of signal has to be followed and then also controlling the traffic lights based on the implementation model selected.
* User in the vehicle checks the traffic lights and based on those traffic lights the vehicle driver will move forward or wait for the signal to turn green.
* All the collected and analyzed data by the traffic controller will then be stored in the MUMS Database as a log or for further reference.
* Traffic Controller will also send the required data to the DCC in case of any accidents or if any driver breaks the traffic signal indicating that the DCC will have to look into the issue.
* In the implementation mode, a Computation algorithm (Compute Engine) is also added that interacts with the implementation mode and will make sure that the traffic lights are controlled based on the algorithm. For example, the traffic control system will behave differently during the day and will behave differently during the night.
* DCC can also access the MUMS Database as and when required.
* DCC can also manipulate the Computational Algorithm as and when required.

A screenshot of a cell phone

Description automatically generated**Figure 2.**

* When the EVAS Vehicle is called for a separate trigger is activated.
* When an EVAS system is started it would send an event called “START” to the traffic controller and once the event is started the controller would activate the EVAS Vehicle Tracker and would be checking the distance from the vehicle to the next signal.
* Once the Vehicle is less than 100 meters away from the signal it would immediately turn the signal color to green to clear away any fallen back traffic and also change the other signals to red to make way for the EVAS Vehicle to pass by.
* After the EVAS Vehicle is passed the signal would go back to its normal settings after 20 seconds

**A close up of text on a white background

Description automatically generatedFigure 3.**

1. The signal (red in color) would first beep the vehicle to check if it is an EVAS Vehicle.
2. If YES, the color would change itself to green making way for the EVAS Vehicle to pass by.
3. If NO, then the signal records of the vehicles approaching and also sends an analyzed data to DCC.
4. If there are any Violations happening in the signal the DCC would send immediate instructions to the signal to take measures (example: dropping a ticket for a car if a signal is jumped) using the registration number of the vehicle.
5. If there are no violations, then the signal would be running in the normal state and the timer would still be running on the signal.
6. Incase there is a sudden emergency the DCC would take full control over the signal and will adopt it according to the special measures that needs to be taken.

**A close up of text on a white background

Description automatically generatedFigure 4.**

* There are 7 tables that are involved and each of these tables hold unique details to complete the whole architecture.
* Official Role has entities like role\_id that will hold the role for a particular user in the system. The user in that specific role will also hold a specific title associated to him/her.
* User Class has entities like userid, roleid, name etc. This table will hold the complete details of the user that will be accessing the system and based on the rolled he will be allotted permissions on the system.
* Permission Class will store the details on what all permission a particular role will have on the system, functions like adding a new permission, editing a permission and deleting a permission can be carried out on this table.
* DCC will have permission to add a traffic controller, monitor the password, set the username etc. DCC will have complete access over the permission class.
* Diversion will have details like the diversion type, Diversion name etc. Over here the diversion type can indicate how feasible the diversion can be to vehicles. Functions like search diversion, add diversion, delete a diversion, edit a diversion can be carried out here.
* Route Class will have details about the complete route and will indicate the length in miles for the resulting route. It will also hold the route name along with the route type. Route type over here indicates the traffic condition of the route.
* Traffic Class will hold details about the current traffic situation, traffic description will tell more details about the current scenario of the traffic and the time in this entity will indicate the time interval for changing a signal 2.

**SYSTEM ARCHITECTURE**

A close up of text on a white background

Description automatically generated

For a system as critical as a traffic control system, we shall be using repository architecture because all the data getting processed is through Dublin City Council Data Centre. The system has various subsystems that is, Traffic Controller, MUMPS Database, Congestion Algorithm, which communicate with each other. The DCC Data centre acts as a central repository for all the data and can be accessed by al the subsystems. The various subsystems co-depend on each other through DCC Data Centre. The data from the Camera and road sensors are sent to the DCC Data Centre as live feed and simultaneously sent to the Database and Congestion Algorithm as well for analysis. The purpose of using this architectural model is that we will be handling large amounts of data which need to be processed and manipulated in a smooth and efficient manner. All components are independent of each other and they are their own system. Any changes we make to any of the sub-systems like Traffic controllers or Cameras or Road sensors will not affect the whole system but, at the same time they can be easily synchronized and integrated. The data from the traffic controller is sent to DCC data centre and then congestion algorithm, where a vast collection of data is analysed, and appropriate de-congestion input files are sent over to their respective signal ID’s. All through this the interface will remain independent of the implementation since we will be using Bridge Pattern.

Bridge pattern is the best design pattern to be implemented here, since, it keeps the interface to the sub-systems constant while allowing us to change the actual class and display of the DCC data centres viewing applications. We can even hide the implementation procedure from the users of Datacentre during software changes.

**CODE:**

**FILE 1:** signalid 1 time 2 colorfrom red colorto green evasoverride 1

**FILE 2:** signalid 1 time 2 colorfrom green colorto amber evasoverride0

**FILE 3:** signalid 1 time 2 colorfrom amber colorto red evasoverride 1

**OUTPUT FILE:** 1 evasoverride green colorto red colorfrom 2 time 1 signalid 0 evasoverride amber colorto green colorfrom 2 time 1 signalid

**ERROR LOG FILE:** 1 evasoverride red colorto amber colorfrom 2 time 1 signalid

**PYTHON CODE:**

f=open("file1.txt",mode='r')

quizlist=""

space=" "

QuizList=[]

for line in f:

QuizList.append(line.split())

print(QuizList)

for i in range(0,len(QuizList)):

for j in range(0,10):

if QuizList[i][j]=="signalid":

print(QuizList[i][j+1])

signalid=QuizList[i][j+1]

if QuizList[i][j]=="time":

print(QuizList[i][j+1])

time=QuizList[i][j+1]

if QuizList[i][j]=="colorfrom":

print(QuizList[i][j+1])

colorfrom=QuizList[i][j+1]

if QuizList[i][j]=="colorto":

print(QuizList[i][j+1])

colorto=QuizList[i][j+1]

if QuizList[i][j]=="evasoverride":

print(QuizList[i][j+1])

evas=QuizList[i][j+1]

for i in range(0,len(QuizList)):

for j in range(0,10):

quizlist=QuizList[i][j]+space+quizlist

if evas=="1":

if colorto=="green":

print("correct")

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

else:

b=open("errorlogfile.txt",mode='a')

b.write(quizlist)

b.close()

else:

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

f.close()

f=open("file2.txt",mode='r')

quizlist=""

space=" "

QuizList=[]

for line in f:

QuizList.append(line.split())

print(QuizList)

for i in range(0,len(QuizList)):

for j in range(0,10):

if QuizList[i][j]=="signalid":

print(QuizList[i][j+1])

signalid=QuizList[i][j+1]

if QuizList[i][j]=="time":

print(QuizList[i][j+1])

time=QuizList[i][j+1]

if QuizList[i][j]=="colorfrom":

print(QuizList[i][j+1])

colorfrom=QuizList[i][j+1]

if QuizList[i][j]=="colorto":

print(QuizList[i][j+1])

colorto=QuizList[i][j+1]

if QuizList[i][j]=="evasoverride":

print(QuizList[i][j+1])

evas=QuizList[i][j+1]

for i in range(0,len(QuizList)):

for j in range(0,10):

quizlist=QuizList[i][j]+space+quizlist

if evas=="1":

if colorto=="green":

print("correct")

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

else:

b=open("errorlogfile.txt",mode='a')

b.write(quizlist)

b.close()

else:

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

f.close()

f=open("file3.txt",mode='r')

quizlist=""

space=" "

QuizList=[]

for line in f:

QuizList.append(line.split())

print(QuizList)

for i in range(0,len(QuizList)):

for j in range(0,10):

if QuizList[i][j]=="signalid":

print(QuizList[i][j+1])

signalid=QuizList[i][j+1]

if QuizList[i][j]=="time":

print(QuizList[i][j+1])

time=QuizList[i][j+1]

if QuizList[i][j]=="colorfrom":

print(QuizList[i][j+1])

colorfrom=QuizList[i][j+1]

if QuizList[i][j]=="colorto":

print(QuizList[i][j+1])

colorto=QuizList[i][j+1]

if QuizList[i][j]=="evasoverride":

print(QuizList[i][j+1])

evas=QuizList[i][j+1]

for i in range(0,len(QuizList)):

for j in range(0,10):

quizlist=QuizList[i][j]+space+quizlist

if evas=="1":

if colorto=="green":

print("correct")

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

else:

b=open("errorlogfile.txt",mode='a')

b.write(quizlist)

b.close()

else:

a=open("logfile.txt",mode='a')

a.write(quizlist)

a.close()

f.close()