**Assignment 1**

Thi Van Anh DUONG Student ID: 90023112

Diploma of Information Technology, Curtin College

ISEN1000: Introduction to Software Engineering

Coordinator: Khurram Hameed

18 April 2022 **Student Declaration of Originality**

|  |  |
| --- | --- |
|  | This assignment is my own original work, and no part has been copied from another student’s work or source, except where it is clearly attributed. |
|  | All facts/ideas/claims are from academic sources are paraphrased and cited/referenced correctly. |
|  | I have not previously submitted this work in any form for THIS or for any other unit; or published it elsewhere before. |
|  | No part of this work has been written for me by another person. |
|  | I recognise that should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with Curtin College policy. |

**Electronic Submission:**

|  |  |
| --- | --- |
|  | I accept that it is my responsibility to check that the submitted file is the correct file, is readable and has not been corrupted. |
|  | I acknowledge that a submitted file that is unable to be read cannot be marked and will be treated as a non-submission. |
|  | I hold a copy of this work if the original is damaged, and will retain a copy of the Turnitin receipt as evidence of submission. |

1. **Requirments and Planing**
2. **Functional requirements:**
3. **Actor:**

A software system uses to control multiple buses for: **TRANSPERTH**

* Administrators (Human)
* Passengers (Human)
* Operators (Human)
* Drivers (Human)
* Developers (Human)
* Database (Non-Human)
* Sensors (Non-Human)
* GPS (Non-Human)

1. **User stories:**

* Administrators:
* As an administrator, I want to login to Transperth fanpage so that I can response to user’s feedback.
* As an administrator, I want to add routes so that all passengers can catch their bus in proper time.
* As an administrator, I want to update routes from 6.am to 10p.m so that passengers can plan their traveling.
* As an administrator, I want to remove some routes that having less passengers riding.
* As an administrator, I want to add, change or remove Bus Stations.
* As an administrator, I want to display real time bus location or any delay in their route.
* As an administration, I want to extract information regarding passenger density in a day/week/month so that it can greatly facilitate for further planning and development of the whole system.
* As an administration, I want to add, edit or delete the drivers from drivers’ list in the system so that I can ensure the system’s quality.
* Passengers:
* As a service user, I want plan my journey in Transperth system so that I know when I can catch my bus and know how long does it take.
* As a passenger, I want to save my favorite journey so that I can travel faster.
* As a passenger, I want to pay my ticket via mobile app or travelling card with an automatic account management.
* As a passenger, I want to create an account, login in order to use the bus service.
* As a student (passenger), I want to get discount (student fare) for my ticket so that I can save money.
* As a passenger, I want to give Transperth my feedback or complains so that the whole system can improve their quality better.
* As a passenger, I want to check route information, bus stops, departure time, estimated time of arrival of stops so that I will not miss my bus.
* Operators:
* As an operator, I want to appoint a new driver or more driver so that avoiding the probability of being late in special circumstances (traffic congestion, accidents, bus broken, etc…)
* Drivers:
* As a driver, I want to view my passenger’s feedback regarding to driving skills, attitude so that I can improve it.
* Developers:
* As a developer, I want to design more attributes for the system so that the whole bus system can run smoothly.

1. **Use cases**

***Use case 1: Log in***

* *Goal*: To log in to the system.

*Primary actor*: member (**Administrator, Passenger)**

*Secondary actor*: Database

*Trigger*: Administrator click “log in” button to login to the system.

*Precondition*: member that haven’t login to the system.

*Flow of event:*

1. Member selects ‘Log in’
2. Login form appears.
3. Type user name, password to the login form
4. The system will double check user name, password of the member.
5. If the member types the wrong user name and password the use case will move to extension 1A. Otherwise, continuing performing other features.
6. Use case end.

*Extension:*

1A. Member login unsuccessfully:

1. The system announces that login process unsuccessfully.
2. Select “Sign in” or re-type user name and password. If selecting “Sign in”, continue use case 2: Sign in
3. The system asks member re-type user name and password.
4. If the member agrees, the use case resumes at step 2 (FOE). Otherwise, the use case ends.

***Use case 2: Sign in***

* *Goal*: To sign in as the member of the system.

*Primary actor*: member (**Administrator, Passenger)**

*Secondary actor*: Database

*Trigger*: Administrator click “sign in” button to sign in to the system.

*Precondition*: member that haven’t used to the system before.

*Flow of event:*

1. Member selects ‘Sign in’
2. Sign in form appears.
3. Type necessary personal information to the form.
4. Click ‘Submit’ button.
5. The system announces the result of entering personal information. If member’s information is not true then the use case will move to extension 1A. Otherwise, the use case will continue at step 6.
6. The system updates member’s information to member list.
7. Use case ends.

*Extension:*

1A. Inaccurate information:

1. The system announces that the information is inaccurate.
2. The system asks user check their information again
3. If user agrees, the use case resume at step 2. If not the use case ends.

***Use case 3: Manage drivers***

* *Goal*: To add, edit information or delete drivers from the drivers’ list.

*Primary actor*: **Administrator**

*Secondary actor*: Database

*Trigger*: Administrator click “confirm” to perform the operation.

*Precondition*: log in successfully to the system.

*Flow of event:*

1. The administrator chooses option to do with drivers: add, update or delete drivers from the list.
2. Add drivers:
   1. The system appears form to key the driver’s information.
   2. The Administrator types in the driver’s information
   3. Click “Save” button
   4. If typing successfully, the use case will continue at step 5. Otherwise, the use case will enter extension 1A
   5. Save driver’s information.
3. Edit driver’s information:
   1. The system appears form to edit driver’s information.
   2. The administrator keys all information that needs to update.
   3. Click “Save” button
   4. If save successfully, the use case will move to step 5. Otherwise, the use case will enter extension 1A.
   5. Save driver’s information.
4. Delete driver:
   1. The administrator chooses drivers want to delete.
   2. Click “Delete” to delete driver.
   3. The system ask administrator to confirm deleting. If the administrator agrees, the use case will continue at step 4. Otherwise, the use case will move to step 5.
   4. Announcing that driver has been deleted.
   5. The system displays the updated driver’s list.
5. The use case end.

*Extension:*

1A. Inaccurate information:

1. The system announces that the information is inaccurate.
2. The administrator types information again
3. The use case resumes at step 1.3.

***Use case 4: Remove bus route***

* *Goal*: To remove bus route that have less passengers riding.

*Primary actor*: **Administrator**

*Secondary actor*: Database

*Trigger*: Administrator click “confirm” button to remove the class.

*Precondition*: computer-based knowledge, log in successfully to the system.

*Flow of event:*

1. Administrator gather lists of regions, times, buses that usually have less passenger taking part in.
2. Administrator selects routes he/she wants to delete in the app.
3. Administrator click “remove”.
4. The system ask password for confirm deleting.

*Extension:*

1A. Administrator select the wrong route:

1. The administrator goes back to the old working session.
2. The administrator click “recover” button to put the route being deleted wrong.
3. The use case resumes at step 2.

1B. Administrator forgets his/her password.

1. The system asks administrator select “Forget password” option.
2. The system asks administrator reset password via email or phone number.
3. The use case resumes at step 4.

***Use case 5: Plan a journey***

* *Goal*: To plan a journey.

*Primary actor*: **Passenger**

*Secondary actor*: Database, GPS

*Trigger*: click “OK” button to plan a journey.

*Precondition*: login or create a new account to start using bus services

*Flow of event:*

1. Passenger login or create a new account to begin using the service
2. Passenger key their current position, their desired position, date and time in searching bar.
3. Click “OK” to start find proper journeys.
4. Passenger choose the suitable journey.

*Extension:*

1A. No bus is suitable to passenger’s requirement:

1. The system will appear the message “Oh no, your bus is not available”.
2. The system will recommend passenger the latest route with reasonable time.
3. The passenger chooses their route.
4. The use case end.

1B. The bus was unexpectedly canceled, the departure time was delayed.

1. The system will send passenger a message “Sorry! <content>”
2. Passenger checks bus information.
3. Passenger chooses another bus.
4. The use case resumes at step 2.

***Use case 6: Distribute drivers***

* *Goal*: To distribute drivers.

*Primary actor*: **Operator**

*Secondary actor*: Database, GPS, Drivers

*Trigger*: click “Confirm” button to appoint appropriate drivers*.*

*Precondition*: knowing driver’s schedule.

*Flow of event:*

1. Administrator gather lists of drivers who are free in the time they are going to be appointed.
2. Administrator selects drives he/she thinks suitable.
3. Administrator click “confirm” to start appointing drivers.
4. The system sends a message to drivers about the route information.

*Extension:*

1A. The driver refuses:

1. The system will announce to the operator.
2. The operator will find another driver.
3. If cannot the use case resumes at step 2.
4. The operator will delete this route.

***Use case 7: Manage Bus Station***

* *Goal*: To add, change or remove Bus Station.

*Primary actor*: **Administrator**

*Secondary actor*: Database

*Trigger*: click “Confirm” button to perform the operation*.*

*Precondition*: already log in successfully to the system.

*Flow of event:*

1. The administrator chooses option to do with Bus Station: add, change or remove.
2. Add Bus Station:
   1. The system appears form to key the Bus Station information.
   2. The Administrator types in the Bus Station information
   3. Click “Save” button
   4. If typing successfully, the use case will continue at step 5. Otherwise, the use case will enter extension 1A.
   5. Save place information.
3. Change Bus Station location:
   1. The system appears form to edit place information.
   2. The administrator keys all information that needs to update.
   3. Click “Save” button
   4. If save successfully, the use case will move to step 5. Otherwise, the use case will enter extension 1A.
   5. Save place information.
4. Remove Bus Station:
   1. The administrator chooses place want to delete.
   2. Click “Confirm” to remove the place.
   3. The system ask administrator to confirm removing. If the administrator agrees, the use case will continue at step 4. Otherwise, the use case will move to step 5.
   4. Announcing that place is no longer active
   5. The system displays the updated Bus Station list.

2. The use case end.

*Extension:*

1A. Inaccurate information:

1. The system announces that the information is inaccurate.
2. The administrator types information again.
3. The use case resumes at step 1.3.
4. **Use case diagram**

A software system uses to control multiple buses for Transperth: **TRANSPERTH**



1. **Non-Functional requirements:**
2. **Usability Requirement**

* The software system must show all journey options, estimated departed times, prices for customers on the screen.
* The software system must allow passengers locate their position with the accuracy at least 99% of the time.
* The software system must allow administrator to track the bus.
* The software system must allow drivers to know about the information of their route which they are in charge.
* The software system must allow central control staff export passenger density as PDF.

1. **Performance Requirement**

* Response time: response to the arrival time of the latter bus and departure time of the previous one (to avoid traffic congestion at bus stop).
* Response time: response to the accidental occasion that delay the bus route.
* Response time: keep passenger updated with the changes (late or soon) and update route (additional bus at restricted hours or school holiday, public holiday, etc…)
* Data processing: update up to 100 new bus route per 15 minutes.
* How efficient its use of resources is? Internet connection (Wireless, 3G, 4G, …), GPS, Electrical Blackboard (Showing bus time), bus station.

1. **Reliability Requirement**

* Mean Time To/Between Failure (MTTF/MTBF): The system must have a MTTF at least 1 year
* Availability (AVAIL): The system should be available 99,999% of the time.
* Probability of Failure on Demand (POFOD): The system’s control feature should have a POFOD of 0.1% or less.
* Rate of Occurrence of Failure (ROCOF): The system must not have more than 1 bus is late in every 1000 hours.

1. **Planning**
2. **Work Breakdown**
3. *Investigate* ***Transperth*** *bus management model*
4. *Analyze and design system*
5. *Recruit Employees:*

c.1. Administrators

c.2. Operators

c.3. Drivers

c.4. Developers

1. *Facilities:*

d.1. Buses

d.2. Bus stations

d.2.1. Locations

d.2.2. Parking policies (special events, rush hours, etc…)

1. *Create Database:*

e.1. Driver information

e.2. Route information

e.3. Bus station information

1. *Create sign in and login feature*
2. *Create functional features:*

g.1. Manage drivers

g.1.1. Add drivers’ information

g.1.2. Edit drivers’ information

g.1.3. Remove drivers’ information

g.2. Manage routes

g.2.1. Add route

g.2.2. Edit route

g.2.3. Remove route

g.3. Manage bus stations

g.3.1. Add bus station

g.3.2. Change bus station

g.3.3. Remove bus station

e.4. Get report

1. *Ticket:*

h.1. Payment

h.2. Discount

1. *Journey:*

i.1. Plan a journey

i.2. Check journey

1. *Feedback*
2. *Testing the system*
3. *Integration and maintenance*
4. **Estimation (WEEKS)**
5. *Investigate* ***Transperth*** *bus management model*  8W
6. *Analyze and design system* 4W
7. *Recruit Employees:*

c.1. Administrators 6W

c.2. Operators 3W

c.3. Drivers 3W

c.4. Developers 6W

1. *Facilities:*

d.1. Buses 12W

d.2. Bus stations

d.2.1. Locations 10W

d.2.2. Parking policies (special events, rush hours, etc…) 5W

1. *Create Database:*

e.1. Driver information 4W

e.2. Route information 5W

e.3. Bus station information 6W

1. *Create sign in and login feature* 3W
2. *Create functional features:*

g.1. Manage drivers

g.1.1. Add drivers’ information 2W

g.1.2. Edit drivers’ information 1W

g.1.3. Remove drivers’ information 1W

g.2. Manage routes

g.2.1. Add route 2W

g.2.2. Edit route 1W

g.2.3. Remove route 1W

g.3. Manage bus stations

g.3.1. Add bus station 3W

g.3.2. Change bus station 4W

g.3.3. Remove bus station 2W

e.4. Get report 2W

1. *Ticket:*

h.1. Payment 5W

h.2. Discount 2W

1. *Journey:*

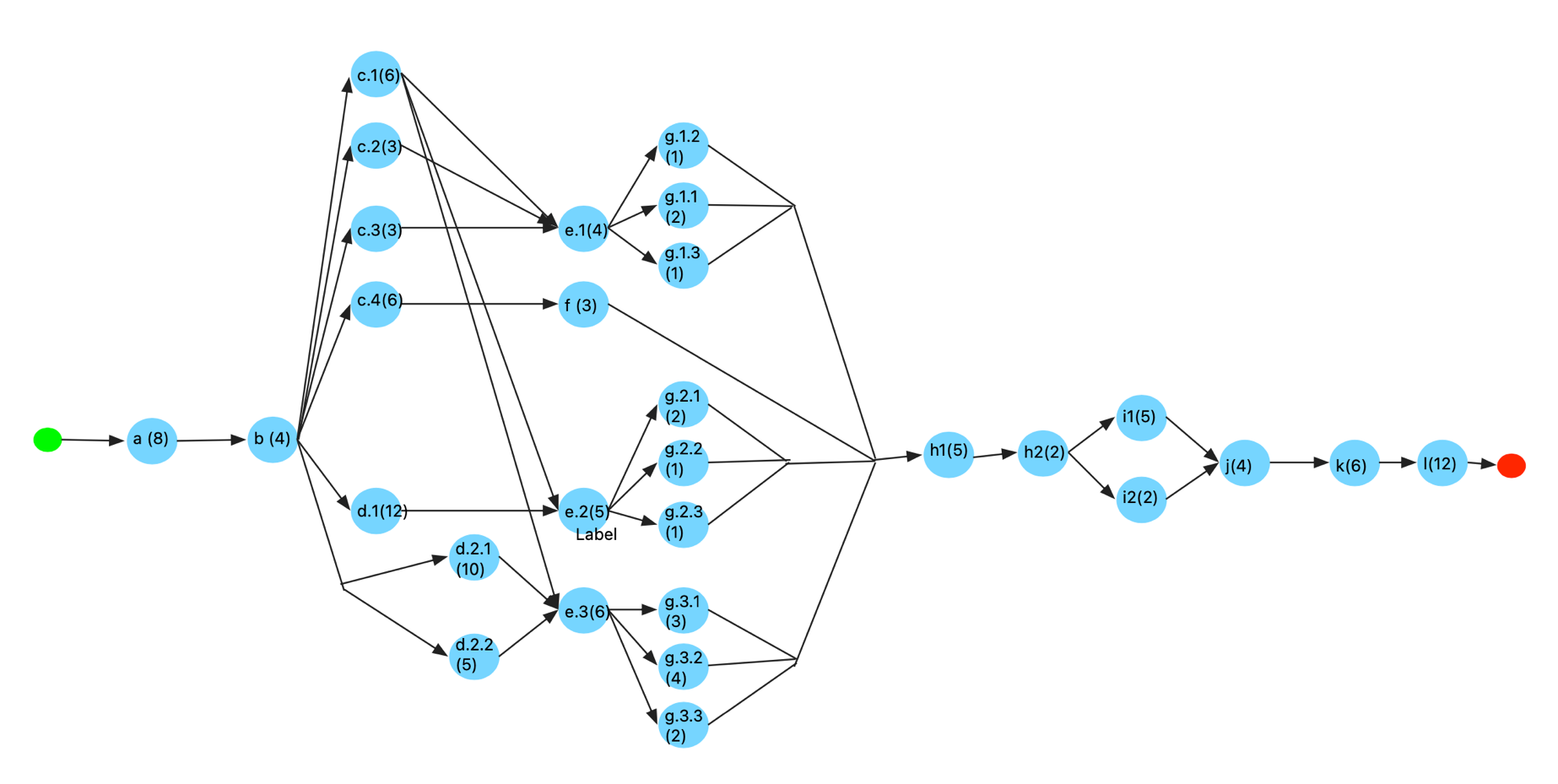
i.1. Plan a journey 1W

i.2. Check journey 1W

1. *Feedback* 4W
2. *Testing the system* 6W
3. *Integration and maintenance* 12W
4. **Prerequisite Activities**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Dependencies** | **Duration** |
| a | - | 8W |
| b | a | 4W |
| c.1 | b | 6W |
| c.2 | b | 3W |
| c.3 | b | 3W |
| c.4 | b | 6W |
| d.1 | b | 12W |
| d.2.1 | b | 10W |
| d.2.2 | b | 5W |
| e.1 | c.1, c.3 | 4W |
| e.2 | c.1, d.1 | 5W |
| e.3 | c.1, d.2 | 6W |
| f | c.4 | 3W |
| g.1.1 | c.2, e.1 | 2W |
| g.1.2 | c.2, e.1 | 1W |
| g.1.3 | c.2, e.1 | 1W |
| g.2.1 | e.2 | 2W |
| g.2.2 | e.2 | 1W |
| g.2.3 | e.2 | 1W |
| g.3.1 | e.3 | 3W |
| g.3.2 | e.3 | 4W |
| g.3.3 | e.3 | 2W |
| h.1 | f, g.1, g.2, g.3 | 5W |
| h.2 | f, g.1, g.2, g.3 | 2W |
| i.1 | h | 5W |
| i.2 | h | 2W |
| J | i | 4W |
| k | j | 6W |
| l | k | 12W |

**Activity-on-Node (AON) Graph With Timing**

****

1. **Critical Path and Slash Time**

Find the earliest start (ES) and earliest finish (EF) for each activity.

* Start with the activities that don’t depend on anything (activity a)
* Work forwards until get to the final activity.
* Early Start Time = 0
* EF = ES + Duration

I have the table as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Duration** | **ES** | **EF** |
| a | 8 | 0 | 8 |
| b | 4 | 8 | 12 |
| c.1 | 6 | 12 | 18 |
| c.2 | 3 | 12 | 15 |
| c.3 | 3 | 12 | 15 |
| c.4 | 6 | 12 | 18 |
| d.1 | 12 | 12 | 24 |
| d.2.1 | 10 | 12 | 22 |
| d.2.2 | 5 | 12 | 17 |
| e.1 | 4 | 18 | 22 |
| e.2 | 5 | 24 | 29 |
| e.3 | 6 | 22 | 28 |
| f | 3 | 18 | 21 |
| g.1.1 | 2 | 22 | 24 |
| g.1.2 | 1 | 22 | 23 |
| g.1.3 | 1 | 22 | 23 |
| g.2.1 | 2 | 29 | 31 |
| g.2.2 | 1 | 29 | 30 |
| g.2.3 | 1 | 29 | 30 |
| g.3.1 | 3 | 28 | 31 |
| g.3.2 | 4 | 28 | 32 |
| g.3.3 | 2 | 28 | 30 |
| h.1 | 5 | 32 | 37 |
| h.2 | 2 | 37 | 39 |
| i.1 | 5 | 39 | 44 |
| i.2 | 2 | 39 | 41 |
| J | 4 | 44 | 48 |
| k | 6 | 48 | 54 |
| l | 12 | 54 | 66 |

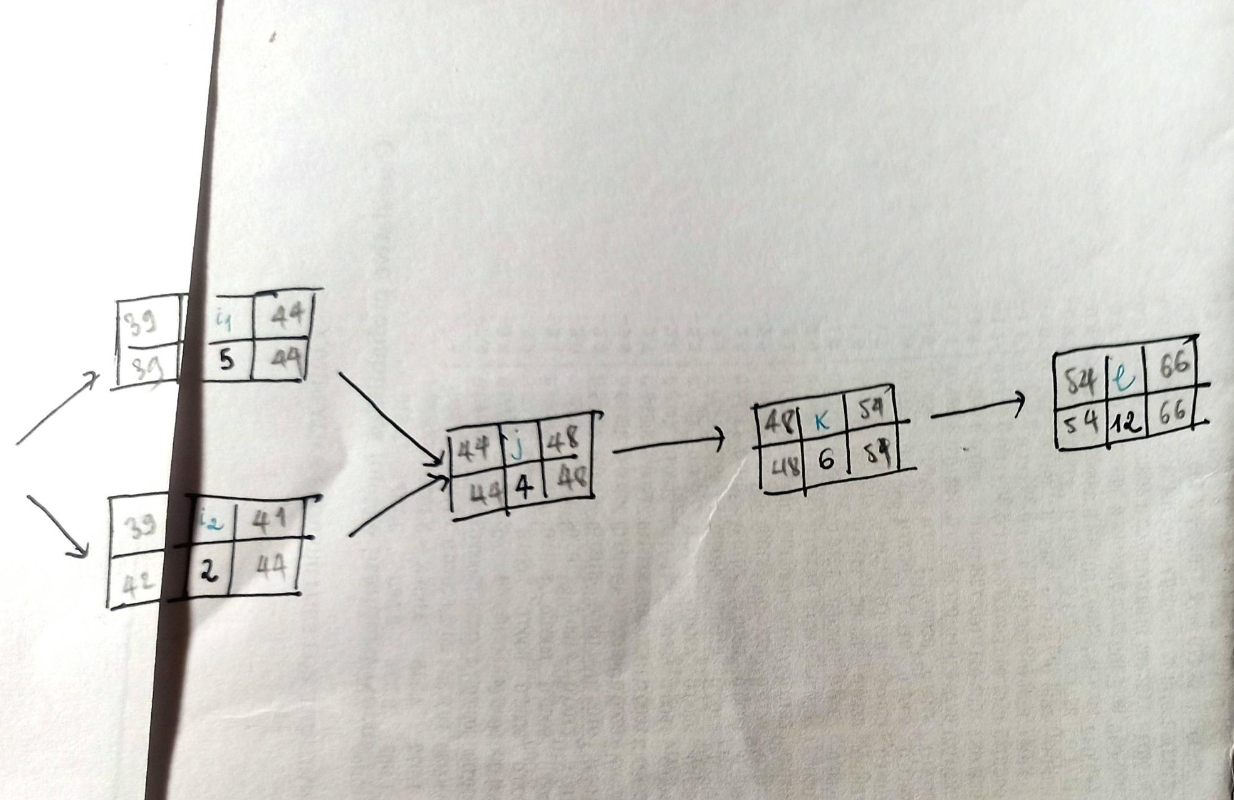
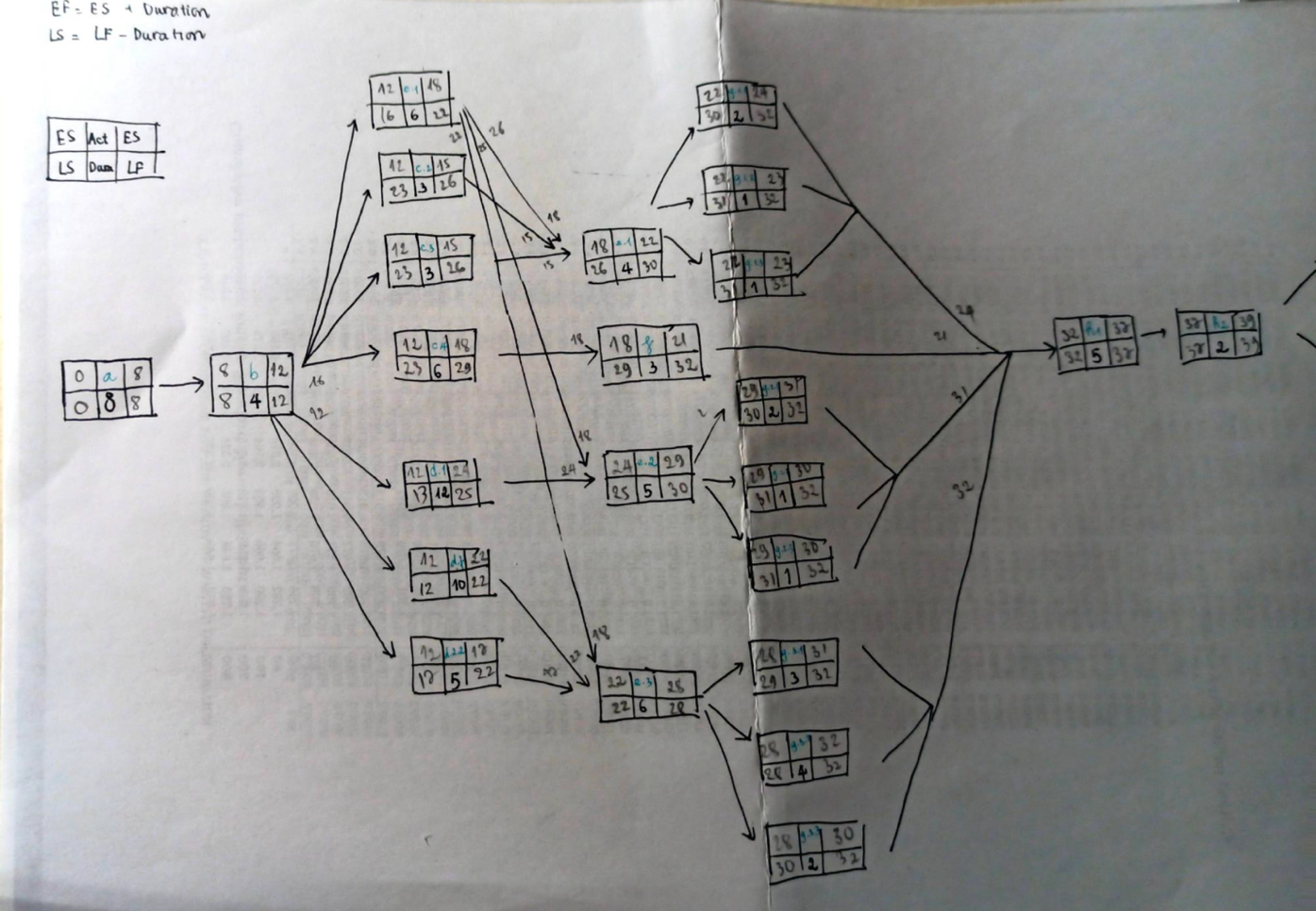
Find the latest start (LS) and latest finish (LF) for each activity.

* Start with the activities at the end of the project
* Latest Finish of the activity l = Earliest Finish of the activity l
* Work backward until get to the first activity
* LS = LF – Duration

I have the table as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Duration** | **LF** | **LS** |
| l | 12 | 66 | 54 |
| k | 6 | 54 | 48 |
| J | 4 | 48 | 44 |
| i.2 | 2 | 44 | 42 |
| i.1 | 5 | 44 | 39 |
| h.2 | 2 | 39 | 37 |
| h.1 | 5 | 37 | 32 |
| g.3.3 | 2 | 32 | 30 |
| g.3.2 | 4 | 32 | 28 |
| g.3.1 | 3 | 32 | 29 |
| g.2.3 | 1 | 32 | 31 |
| g.2.2 | 1 | 32 | 31 |
| g.2.1 | 2 | 32 | 30 |
| g.1.3 | 1 | 32 | 31 |
| g.1.2 | 1 | 32 | 31 |
| g.1.1 | 2 | 32 | 30 |
| f | 3 | 32 | 29 |
| e.3 | 6 | 28 | 22 |
| e.2 | 5 | 30 | 25 |
| e.1 | 4 | 30 | 26 |
| d.2.2 | 5 | 22 | 17 |
| d.2.1 | 10 | 22 | 12 |
| d.1 | 12 | 25 | 13 |
| c.4 | 6 | 29 | 23 |
| c.3 | 3 | 26 | 23 |
| c.2 | 3 | 26 | 23 |
| c.1 | 6 | 22 | 16 |
| b | 4 | 12 | 8 |
| a | 8 | 8 | 0 (equal to project duration) |

**Here is the diagram that I use to calculate ES, EF, LS, LF.**

****

Slash Time = LF – EF = LS – ES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Activity** | **Duration** | **ES** | **EF** | **LF** | **LS** | **Slash Time** |
| a | 8 | 0 | 8 | 8 | 0 | 0 |
| b | 4 | 8 | 12 | 12 | 8 | 0 |
| c.1 | 6 | 12 | 18 | 22 | 16 | 4 |
| c.2 | 3 | 12 | 15 | 26 | 23 | 11 |
| c.3 | 3 | 12 | 15 | 26 | 23 | 11 |
| c.4 | 6 | 12 | 18 | 29 | 23 | 11 |
| d.1 | 12 | 12 | 24 | 25 | 13 | 1 |
| d.2.1 | 10 | 12 | 22 | 22 | 12 | 0 |
| d.2.2 | 5 | 12 | 17 | 22 | 17 | 5 |
| e.1 | 4 | 18 | 22 | 30 | 26 | 8 |
| e.2 | 5 | 24 | 29 | 30 | 25 | 1 |
| e.3 | 6 | 22 | 28 | 28 | 22 | 0 |
| f | 3 | 18 | 21 | 32 | 29 | 11 |
| g.1.1 | 2 | 22 | 24 | 32 | 30 | 8 |
| g.1.2 | 1 | 22 | 23 | 32 | 31 | 9 |
| g.1.3 | 1 | 22 | 23 | 32 | 31 | 9 |
| g.2.1 | 2 | 29 | 31 | 32 | 30 | 1 |
| g.2.2 | 1 | 29 | 30 | 32 | 31 | 2 |
| g.2.3 | 1 | 29 | 30 | 32 | 31 | 2 |
| g.3.1 | 3 | 28 | 31 | 32 | 29 | 1 |
| g.3.2 | 4 | 28 | 32 | 32 | 28 | 0 |
| g.3.3 | 2 | 28 | 30 | 32 | 30 | 2 |
| h.1 | 5 | 32 | 37 | 37 | 32 | 0 |
| h.2 | 2 | 37 | 39 | 39 | 37 | 0 |
| i.1 | 5 | 39 | 44 | 44 | 39 | 0 |
| i.2 | 2 | 39 | 41 | 44 | 42 | 3 |
| J | 4 | 44 | 48 | 48 | 44 | 0 |
| k | 6 | 48 | 54 | 54 | 48 | 0 |
| l | 12 | 54 | 66 | 66 | 54 | 0 |

**Critical Path**: **a 🡪 b 🡪 d.2.1 🡪 e.3 🡪 g.3.2 🡪 h1 🡪 h2 🡪 i1 🡪 j 🡪 k 🡪 l.**

In this critical path, each activity is a critical activity. If I delay any of them, I will delay the whole project.

Therefore, in order to ensure the progress of the bus management software system, I have to always monitor the progress of critical activities ( because it will tell me whether the project is on track or not).

1. **A Plan for the use of version control**
2. **Version Control**