COMP2050 Assignment 2

Project A

From: Prithivi Sharma, 45705704, Partner(Edoardo Busano, 45757100)

Submitted To: Michael Johnson, COMP2050

Vision Statement

- The vision of this project is to ensure the safety of passengers who use trains as a means for transportation on a daily basis through the practice of social distancing when they board a train. As a result, this will reduce chances of passengers falling sick and having illness in the future.

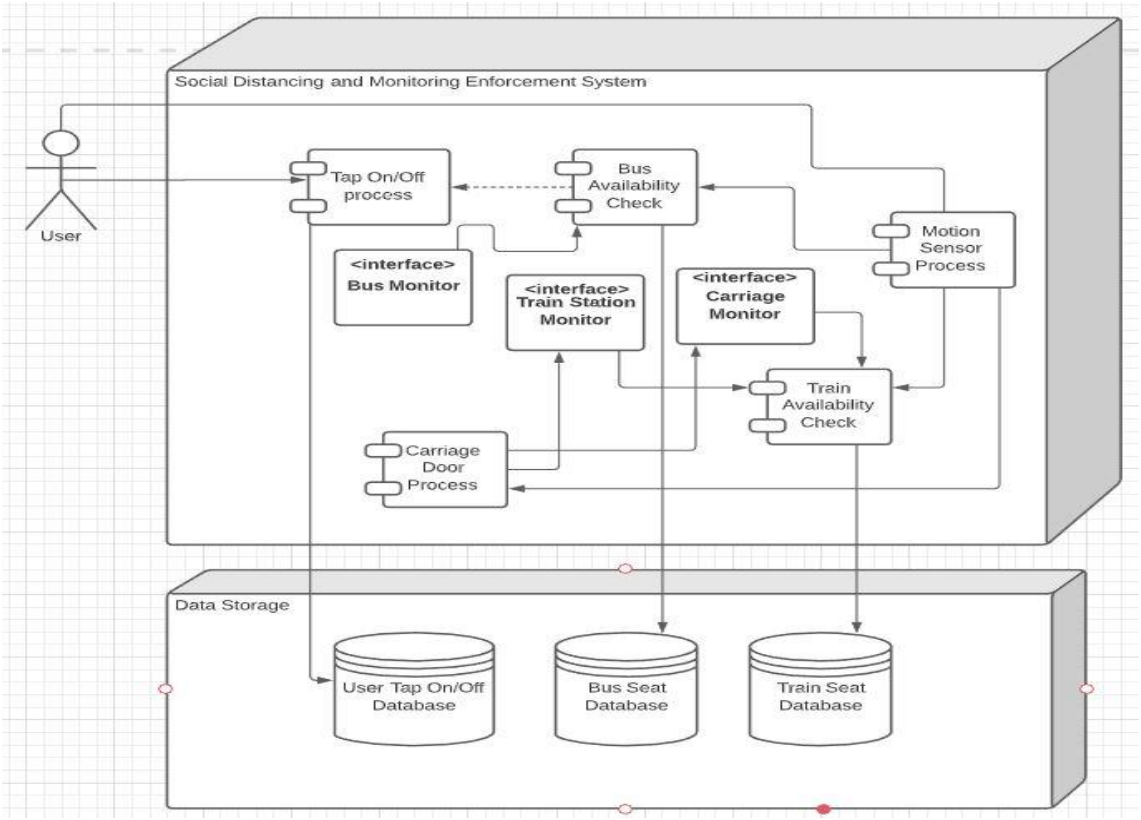
System Design Document

Social Distancing Monitoring And Enforcement System For Transport NSW

Date: 10/10/2020

Author: Prithivi Sharma, 45705704

System Architecture



Storage Data Strategy

- The strategy in storing data within the software system in this context will focus on real-time data capture and transmission and its security. This strategy was considered in contrast other data storing strategies as this form of strategy is more suitable to this context and is better able to meet the requirements set for this system. This is evident from the different processes within the implementation of the software system as they require the use of real-time data in order to make sure that the main requirements are put into place and are being used in effective order. Another reason why the strategy of using real-time data is considered for this project is due to the nature of this project which requires data to be passed with a high level of frequency and speed and this can be only done with effectiveness from the use of real-time data extraction from the software system.

Noteworthy trade-offs and choices

- The choices towards the successful and maintained implementation of the requirements for this software system were focused on the prioritisation of the safety of users in comparison to increased use of trains by users. This is evident from the choice of ensuring that every carriage can hold up to only 16 users with 8 users to be placed on each berth of the carriage. This would trade-off over-crowding within train carriages and ensure that social distancing is maintained within users who would use trains. This same choice is also replicated in the reduction of the number of users who would also use buses when they are in use. This was also decided in order to ensure that social distancing is not only encouraged but practiced on a regular basis by users. This in turn, ensures their complete safety while also ensuring that the main requirements of social distancing are met. Another choice which was also determined during this phase was an increased number of trains and buses being used during daytimes. This choice was decided upon in order to ensure consistent functionality of the system while meeting needs of users and taking the requirements of the systems into consideration as well. This was mainly because of how many users rely upon Opal to meet their transportation needs. From this focused choice, this ensures that the methods of the software system are steady, and it is integrated in unison to the daily use

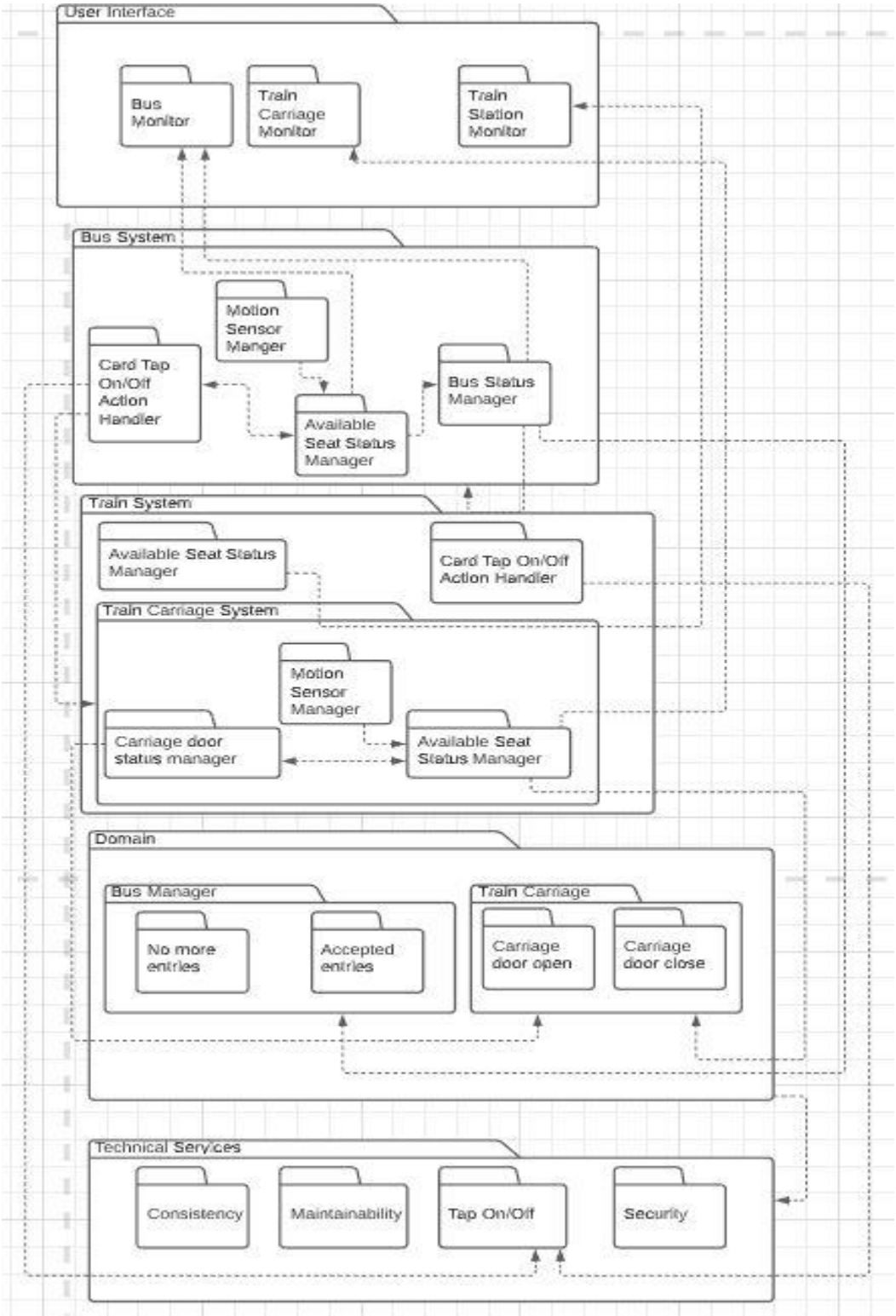
of transportation and making sure that social distancing is maintained within users, allowing the system to be properly maintained accordingly in the future.

Concurrent processes and how they will be coordinated

- One of the concurrent processes that will be used to successfully implement this software system is the real time analysis of seats in a bus or the carriages of a train through motion sensors. This process will allow the system to be able to ensure that the number of users to be placed in each carriage or each bus are accurate and adhere to the rule of the successful practice of maintaining social distancing when using transportation. This process will be coordinated with the transportation method of a train as well as train stations with the motion sensors being able to take in real-time data and transmitting it to systems of train stations and the train, allowing users to be able to know where available seats are located within a train through their carriage monitors as well as screen monitors in the train station, allowing users to be better familiarised with the use of the system and to ensure their safety when using public transportation while also making sure that the rule of maintaining social distancing is used. This will be done so by the system being able to use another process coordinated with this process in the context of trains as carriages that have the maximum number of users will not open their doors once they stop at a train station and only the carriages that do have available seats will open for users to get in accordingly based on the number of seats which will be available. This process will also be coordinated with the public transportation method of bus as well with this method by being able to use motion sensors to be able to ensure that the bus will be able to ensure that the number of users who use buses will not exceed the limit which is required. Through the use of motion sensors, real-time data will also be gathered and sent to bus monitors, allowing users to know the number of available seats there are. This process will be enforced on buses by the software system by not accepting tap-ons of users who are trying to get into a bus which does not have any available seats. Another concurrent process of the software system that is also integrated with its use in trains is the use of motion sensors alongside cameras of each carriage. This process will be used to ensure the correct number of users get into each platform and avoid chances of overcrowding. They will be used to get a real-time indication

of the number of users boarding a train and ensure that the threshold of the number of users of each users for each carriage is not exceeded, therefore meetings its major requirement.

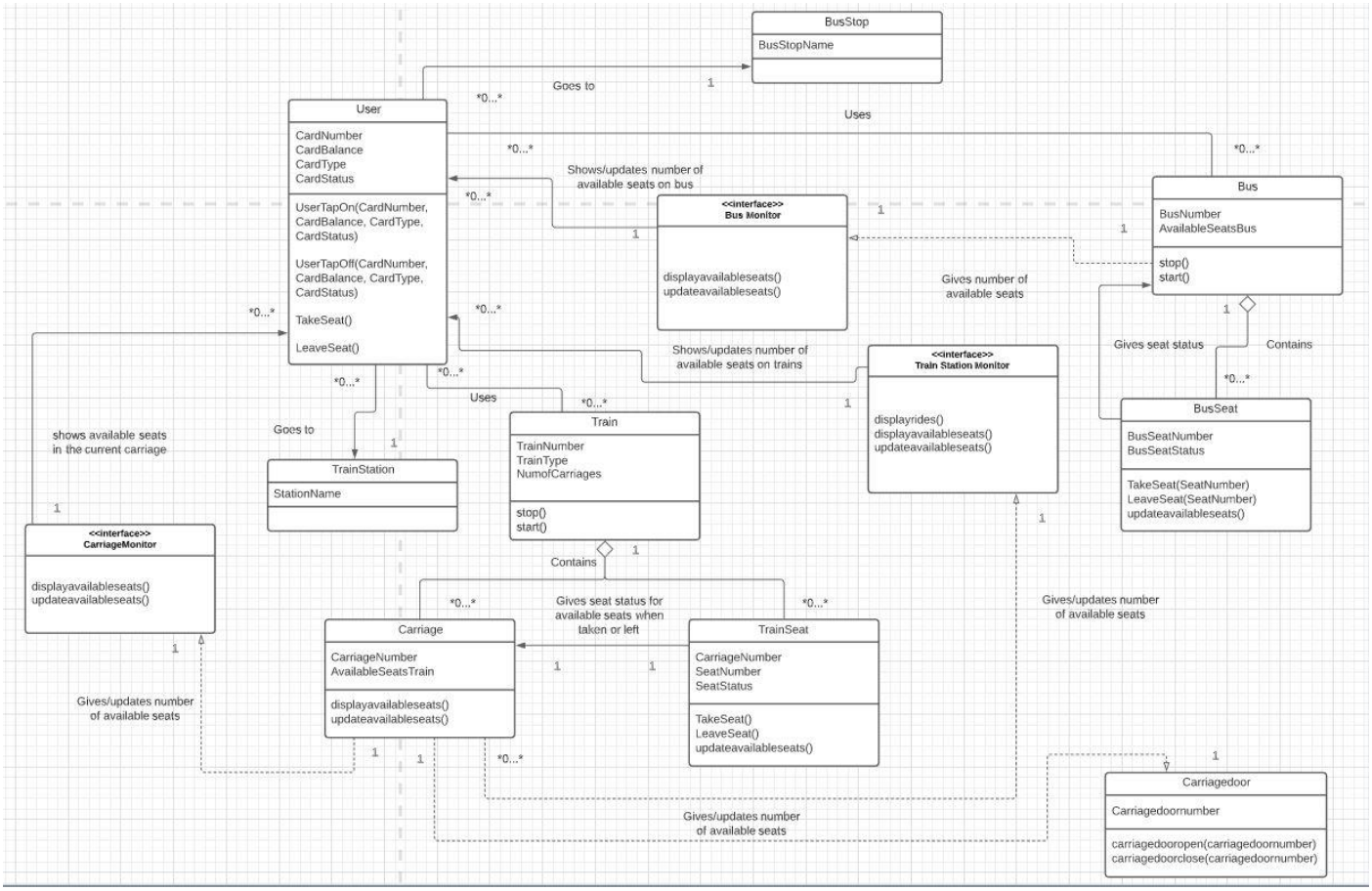
Package Diagram



Data Definitions

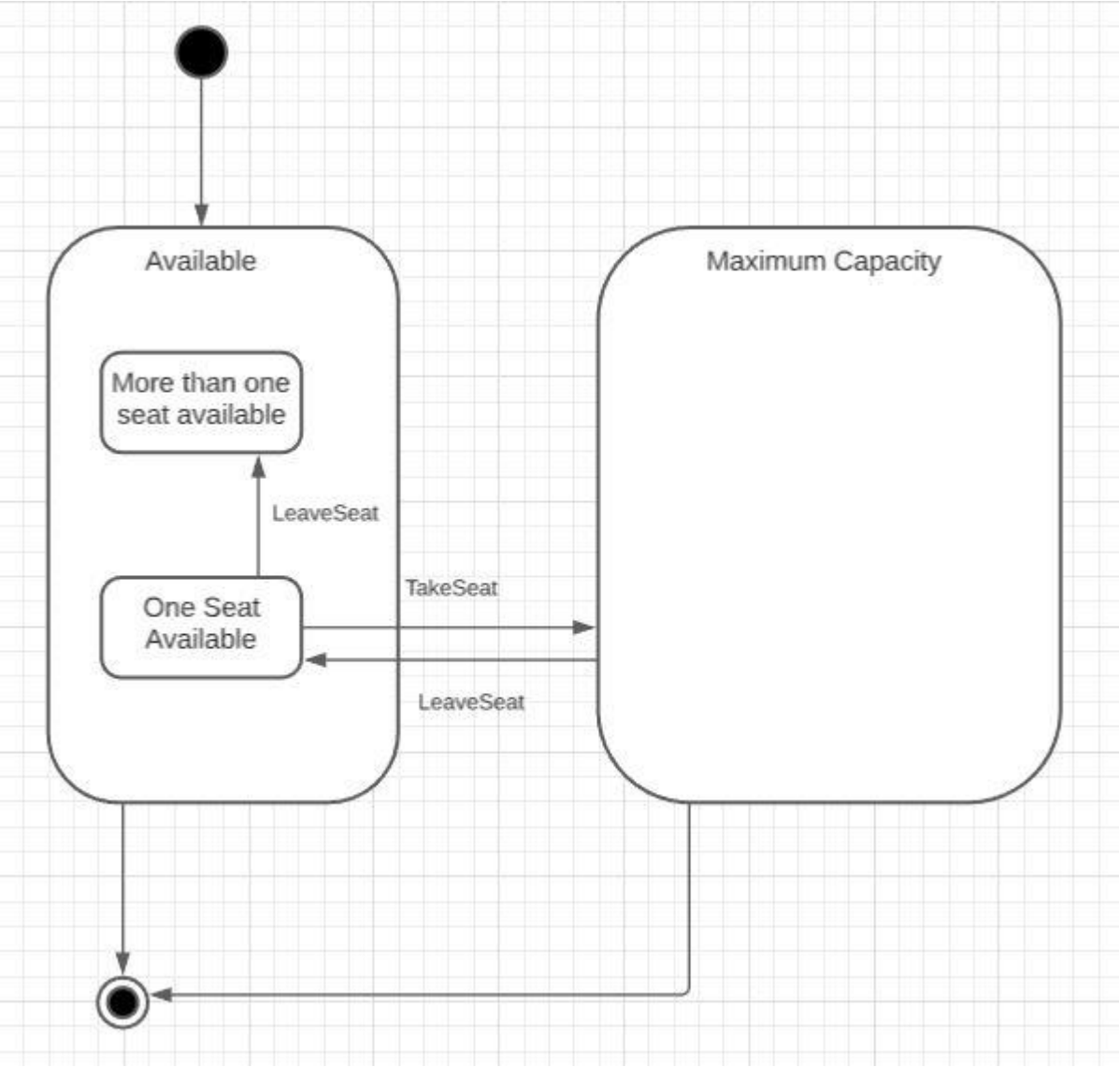
Field	Attribute	Value
User	CardNumber	5423 6578 8890 8767
User	CardBalance	10
User	CardType	Adult
User	CardStatus	Active
Train Station	StationName	Redfern
Train	TrainNumber	L1
Train	TrainType	Express
Train	NumofCarriages	8
Carriage	CarriageNumber	4
Carriage	AvailableSeatsTrain	3
Carriagedoor	Carriagedoornumber	5
TrainSeat	CarriageNumber	2
TrainSeat	SeatNumber	10
TrainSeat	SeatStatus	Occupied
BusStop	BusStopName	Sydney Olympic Park
Bus	BusNumber	255
Bus	AvailableSeatsBus	5
BusSeat	BusSeatNumber	8
BusSeat	BusSeatStatus	Free

Analysis and Design Class Diagram



State Diagram

Object: Bus



Requirement Traceability Matrix

Requirement ID	Use Cases	Classes	Methods	Packages	Build Number
R1	- Tap On	User	UserTapOn(CardNumber,CardBalance,CardType,CardStatus)	User-UserTapOn(CardNumber,CardBalance,CardType,CardStatus)	
R2	- Tap Off	User	UserTapOff(CardNumber,CardBalance,CardType,CardStatus)	User-UserTapOff(CardNumber,CardBalance,CardType,CardStatus)	
R3	- Tap On. - Access denied for maximum capacity.	User Bus	UserTapOn(CardNumber,CardBalance,CardType,CardStatus)	User-UserTapOn(CardNumber,CardBalance,CardType,CardStatus) Bus	
R4	- Display Available Seats.	User Bus Monitor	displayavailableseats	Bus Monitor-displayavailableseats	
R5	- Access denied for maximum capacity reached.	User Carriage Carriagedoor	carriagedoorclose(carriagedoornumber)	Carriagedoor-carriagedoorclose(carriagedoornumber)	
R6	- Take A Seat - Update Available Seats	User TrainSeat CarriageMonitor Carriagedoor	TakeSeat updateavailableseats displayavailableseats carriagedooropen(carriagedoornumber)	User-TakeSeat TrainSeat-updateavailableseats CarriageMonitor-displayavailableseats Carriagedoor-carriagedooropen(carriagedoornumber)	
R7	- Leave Seat - Update Available Seats	User TrainSeat CarriageMonitor Carriagedoor	LeaveSeat updateavailableseats displayavailableseats carriagedooropen(carriagedoornumber)	User-LeaveSeat TrainSeat-updateavailableseats CarriageMonitor-displayavailableseats Carriagedoor-carriagedooropen(carriagedoornumber)	
R8	- Display Available Seats - Update Available Seats	CarriageMonitor	displayavailableseats	CarriageMonitor-displayavailableseats	

R9	- Display Available Seats	CarriageMonitor	displayavailableseats	CarriageMonitor-displayavailableseats	
R10	- Tap On - Display Available Seats - Tap Off	User Bus Monitor Train Station Monitor CarriageMonitor	UserTapOn(CardNumber,CardBalance,CardType,CardStatus) displayavailableseats updateavailableseats	User-UserTapOn(CardNumber,CardBalance,CardType,CardStatus) Bus Monitor-displayavailableseats updateavailableseats Train Station Monitor-displayavailableseats updateavailableseats CarriageMonitor-displayavailableseats updateavailableseats	
R11	- Access Denied for Maximum Capacity Reached - Display Available Seats - Update Available Seats	Carriagedoor Bus Monitor Train Station Monitor CarriageMonitor	displayavailableseats updateavailableseats carriagedoorclose(carriagedoornumber)	Bus Monitor-displayavailableseats updateavailableseats Train Station Monitor-displayavailableseats updateavailableseats CarriageMonitor-displayavailableseats updateavailableseats Carriagedoor-carriagedoorclose(carriagedoornumber)	
R12	- Tap On - Tap Off	User	UserTapOn(CardNumber,CardBalance,CardType,CardStatus) UserTapOff(CardNumber,CardBalance,CardType,CardStatus)	User-UserTapOn(CardNumber,CardBalance,CardType,CardStatus) UserTapOff(CardNumber,CardBalance,CardType,CardStatus)	
R13	- Update Available Seats	Bus Monitor Train Station Monitor CarriageMonitor	updateavailableseats	Bus Monitor-updateavailableseats Train Station Monitor-updateavailableseats CarriageMonitor-updateavailableseats	

Test Specifications

Test-Case Specification

Identifier	T1
Description	User taps their card on to the card reader in the card reader to use a train or a bus.
Input Specifications	- User - User's Opal Card. - Opal tap on/off machine.

Steps	Procedure	Expected Output
1	User looks for an empty card reader for tapping on their card	User finds a card reader that can be used to tap on their card.
2	User taps on their card on the card reader of the train station or the bus they are boarding.	User has successfully tapped on their card onto the card reader of their train station or bus.

Identifier	T2
Description	User gets onto their desired train
Input Specifications	- User - Train - Available Seats On Train.

Steps	Procedure	Expected Output
1	User looks for the train for them to board.	User finds the train number they would like to board on time.
2	User looks for the platform where the train will be at the station.	User successfully finds the platform where their train will be when they can board the train
3	User looks for available seats in the train they are boarding.	Users find available seats for them to use when entering the train.
4	User walks to the platform where their train will be.	User successfully gets to the platform where their train will stop.
5	User boards their desired train.	User successfully boards their desired train with an available seat for them to use.

Identifier	T3
Description	User gets onto their desired bus
Input Specifications	- User - User's Opal Card. - Bus - Available Seats On Bus

Steps	Procedure	Expected Output
1	User looks for the bus they want to board.	User finds the bus they would like to board on time.
2	User looks for available seats in the bus they are boarding.	Users find available seats for them to use in the bus.
3	User taps on their opal card on the card reader of the bus they are boarding.	User successfully taps on their card and is allowed to go in the bus.
4	User makes their way to an available seat in their bus.	User successfully boards their desired bus and gets an available seat for them to use.

Identifier	T4
Description	Available seats on trains are updated once a user leaves
Input Specifications	- User - Train - Available Seats On Train

Steps	Procedure	Expected Output
1	User proceeds to leave the available seat that can be used by users in order to drop off at the train station they intend to drop off.	The opal database receives the data of the available seat for the train.
2	Another user looks for available seats for the train where another user is about to drop off in their monitor.	The number of available seats for the train is updated on the monitor in the train station displaying details of the train they are about to board.

Identifier	T5
Description	Available seats on buses are updated once a user leaves
Input Specifications	- User - Bus - Available Seats On Bus

Steps	Procedure	Expected Output
1	User proceeds to leave the available seat that can be used by users in order to drop off at the bus stop they intend to drop off.	The opal database receives the data of the available seat for the train.
2	Another user looks for available seats for the bus where another user is about to drop off from the bus stop monitor.	The number of available seats for the bus is updated on the monitor in the bus displaying details of the train they are about to board in the bus stop.

Identifier	T6
Description	Available seats on train carriages are updated once a user leaves
Input Specifications	- User - Train - Available Seats On Train

Steps	Procedure	Expected Output
1	User proceeds to leave the available seat that can be used by users in order to drop off at the train station they intend to drop off.	The opal database receives the data of the available seat for the train and its carriages.
2	Another user looks for available seats for the carriage they are currently at in the train.	The number of available seats for the train carriage is updated on the carriage monitor displaying the number of seats that are available for the user to use.

Identifier	T7
Description	Users are not allowed into a bus if it does not contain any available seats
Input Specifications	- User - Bus

Steps	Procedure	Expected Output
1	User taps on their card on a bus that does not have any available seats.	The software system does not accept the tap on of the user and does not allow them on the bus.

Identifier	T8
Description	Users aren't allowed into a train if it does not contain any available seats.
Input Specifications	- User - Train

Steps	Procedure	Expected Output
1	User looks for the train for them to board.	User finds the train number they would like to board on time.
2	User looks for the platform where the train will be at the station.	User successfully finds the platform where their train will be when they can board the train
3	User looks for available seats in the train they are boarding.	The train monitor in the platform shows that there are no available seats on the train.
4	User waits for their train.	The train does not allow users in after they stop at the platform.

Test Plan

Test Schedule

Test Case Identifier	Test Case description	Start Date	End date
T1	User taps their card on to the card reader in the card reader to use a train or a bus.	29/09/2020	30/09/2020
T2	User gets onto their desired train	30/09/2020	2/10/2020
T3	User gets onto their desired bus	2/10/2020	3/10/2020
T4	Available seats on trains are updated once a user leaves	3/10/2020	5/10/2020
T5	Available seats on buses are updated once a user leaves	5/10/2020	6/10/2020
T6	Available seats on train carriages are updated once a user leaves	8/10/2020	15/10/2020
T7	Users are not allowed into a bus if it does not contain any available seats	16/10/2020	18/10/2020
T8	Users aren’t allowed into a train if it does not contain any available seats.	19/10/2020	24/10/2020

Test resourcing

- Hardware

Machine	Configuration
Bus Monitor	The bus monitor is a test resource that is required to ensure that the correct number of available seats are shown when the bus is working.
Carriage Monitor	The carriage monitor is a test resource that is required to show the number of available seats for a carriage
Train Station Monitor	The train station monitor is a test resource that is needed to show users waiting for their desired train the number of available seats.
Carriage Door	The carriage door is a test resource that is needed to determine whether a user is allowed to go inside a particular carriage when a bus stops at a particular station.
Opal Card Reader	The Opal Card Reader is a hardware test resource that is required for users to be able to get into a train or a bus.

- Software

Machine	OS	Software used for testing
Bus Monitor	Motion Sensor Software	Opal Card Reading Software Motion Sensor Software
Carriage Monitor	Motion Sensor Software	Motion Sensor Software
Train Station Monitor	Motion Sensor Software	Motion Sensor Software Train Navigation Software
Carriage Door	Motion Sensor Software	Motion Sensor Software

- Tools

Tools Required	Contact
Opal Card	Opal
Train	RTA
Bus	RTA

Test Milestones

Deliverable	Responsible Party	Milestone/Date
T1	Opal User	1/10/2020
T2	User RTA	3/10/2020
T3	User RTA	4/10/2020
T4	RTA	6/10/2020
T5	RTA	7/10/2020
T6	RTA	16/10/2020
T7	RTA Opal User	19/10/2020
T8	RTA	25/10/2020

Report To Michael

Throughout the process of working on this particular project and further building on its requirements and specifications, I had several problems in working with the SRS provided by my partner. These setbacks included processes to improve the performance and maintainability of the software system through the prioritisation of data storage strategies for the software system, creation of test cases that address and ensure that the functional and the non-functional requirements for the software system are met and the creation of the system architecture in consideration to other factors that can affect the performance of the software product.

The prioritisation of the data storage strategies for the software system is one of the starting problems I encountered when working on this assignment. This was mainly because I had difficulty in determining how data would be taken in and how it be utilised within the software system. The process of prioritising an effective data storage strategy was also difficult as the SRS had not indicated a particular approach towards storing data. Due to that, I also had difficulties in finding approaches within the implementation phase which were suitable for functionality and took the functional and non-functional requirements into consideration. As a result, this delayed my work on other important elements of this project such as the Analysis and Class Diagram as I was unsure on which strategy to apply in the creation of those types of diagrams. I was able to resolve this issue by having a limited form of interactions with other classes where unique data would be stored. This allowed me to better implement other ways of data processing into the implementation process.

The creation of test cases for this software product was one of the most difficult challenges I had during the implementation phase of this project. This was mostly due to the requirements within the SRS addressing different circumstances as well as different entities which were essential in assuring a successful and thorough application of the software system in different types of public transportation in this context. An example of this would be the individual requirements that have been set for public transportation methods of buses and trains. The difference between the structural layout of these types of public transportation would require different test cases set out for each of the areas and methods to increase the likeliness of social distancing being practiced in each of

these modes of transportation. This would result in more test cases with different specifications which put me in a difficult position as I had to consider more factors when developing test cases for this software system, further delaying my implementation process and decreasing chances of building a system which is maintained in the later future. One of the areas where I faced challenges due to this was in the creation of state diagrams and class diagrams as I had to consider different circumstances due to the different contexts and scenarios these two different methods of public transportation. I was able to overcome them by being able to create methods which were able to work within the scope of the software system and were individually able to meet the requirement of social distancing through its use.

The creation of the software architecture for this software system was another problem I had encountered when working on the implementation phase for this software system. This was mainly present because the requirements of the SRS required constant changes within the implementation process for this system where real-time data would be transferred regularly and would need to be stored. This was one of the concerns I had as I had to decide on ways to store that data within the system without creating errors and which classes would be able to inherit and/or manipulate the data if required. This in turn, also affected the strategy I would consider storing data when implementing the software system. I was able to resolve this by creating a limited number of connections with classes where data errors like race conditions could occur and by ensuring that databases which are essential in the effective use and maintenance of the software system did not interact with other classes and methods where data is inherited and collected. As a result, this allowed me to be able to create a robust architecture which successfully met the requirements set for its functionality and can be useful and easy to understand for its users.