ADD SEATS

TEST CASES	OUTPUT
Enter A, passengers are sent from waiting room into the train queue	Entering A each time sends the next set of passengers into the train queue
Random amount generated through a 6-sided dice	Random amounts of passengers are sent each time each is added depending on a 6-sided dice
Once loaded waiting room passengers are displayed	Once loaded, waiting room passengers are displayed along with their name and corresponding seat number
Passengers are added without entering passenger data	Entering of A adds the passengers, no need of entering any kind of details
Train queue displayed with proper arrangements	Upon entering A each time, GUI opens up and the train queue is displayed, with a difference in color, and name and seat number against slot
Display error when queue is full	This particular test case doesn't get to be tested as the queue is assumed to be of size 42. However, once all the passengers have joined the queue entering of A will display the message.
GUI must always be updated	the next time add is called, the train queue is updated

VIEW TRAIN QUEUE

TEST CASES	OUTPUT
Entering of V displays train queue	Enter V on GUI, displays train queue GUI, with 42 slots
Train queue GUI accurate	Each passengers name beside their seat numbers, if they are currently in the queue, else if not, a label saying “empty” is displayed alongside the slot
Train seating arrangement GUI	Shifting scenes to the train seating GUI displays the train seating
Train seating GUI accurate	If passenger has arrived, before joining the queue, his name is labelled beside seat number, and once they enter the train the border color of the button changes signifying on board
Waiting room arrangement GUI	Shifting scenes to the waiting room GUI displays passengers in waiting room
Waiting room GUI accurate	If passenger arrived, they are sent to waiting room, here you can visualize passengers currently in waiting room along with their seat numbers (signified further with red bordering)
GUI must always be updated	Whenever the queue or train has been updated the GUI is updated as well

DELETE PASSENGER FROM QUEUE

TEST CASES	OUTPUT
Entering of D requests user to enter passenger name	Passenger name, NIC and seat number are requested to be entered by user, since even-though, passenger can book multiple seats, still there'll be a different passenger for each seat
Entering of correct details deletes passenger	Corresponding passenger is deleted and their details are shown
Train queue is ordered properly afterwards	Upon deleting of a customer train queue is re-ordered properly, with each passenger, from the deleted passenger's position, brought front
Entering of incorrect name but correct seat number and NIC would still delete	If for instance there is a mix up with same names, in real life, entering of a dummy name along with correct NIC would still delete the corresponding passenger

SAVE DATA

TEST CASES	OUTPUT
Data saved into MongoDB upon entering S	Entering S on the console saves data of the current queue, waiting room, on train, onto the database, along with the instance variables required for the queue class
Upon exiting of program, data must be retained	Data is retained even after closing of the program

LOAD DATA

TEST CASES	OUTPUT
Data is retrieved upon loading	Data saved is brought back into the corresponding data structures and instance variables
GUI is therefore to be updated	GUI will therefore, revert to how it was before closing of program
Any changes done upon loading, saving and retrieval of that data must update the GUI over and over again	The GUI is updated accordingly to the data saved. As the file saved is overwritten each time any changes that have done

RUN SIMULATION

TEST CASES	OUTPUT
Entering R on console runs the simulation	The simulation runs
Passenger currently at the start of the queue added into the train	The passenger who is at the beginning index is added into the train from the queue
A random process delay is generated for each passenger	A random process delay is generated by a total of 3 6-sided die
This process delay is added to all the passengers in the queue	The process delay is added to each passenger's secondsInQueue instance variable
Report of the passenger is generated	A pop-up GUI comes up containing their details, max, least and avg times along with the max length of the queue
The time values and max length is always updated	The values are updated accordingly to the queue passengers
The details of the passengers are to be saved in txt file	The passenger's details, who joined the train, are written into a file writer text file
GUI must always be updated	Upon entering R the queue GUI and on train GUI's of add and view methods are updated
Runs until all passengers in queue have been added into train	Pop-up GUI of each passenger is shown until all passengers have joined the train (the train queue is empty)
Details in the report are accurate	The queue length, max and least times are accurate, as for the average time, it is accurate but is a value rounded down to that of the actual value

VALIDATIONS

TEST CASES	OUTPUT
Prompt display option until Q entered	Menu is continuously prompted until Q is entered
Each option calls a different method	Each option calls a different method. A V calling the add and view methods, S L D R the save, load, delete and run methods respectively
Invalid data type entry/ a wrong option is caught without displaying an error	Any other input other than A, V, S, L, D, R is caught throwing an error message
Closing of GUI during A, V or E doesn't quit program	Clicking on 'X' on the window prompts a confirmation alert to the user, if it was a mistake user can return back into the GUI, if It wasn't the GUI is closed and the menu is called again
Validation for lowercase option entry	The respective methods are called regardless of case-sensitivity.
In delete entering of incorrect details not in the data structure mustn't throw an error.	An appropriate message is outputted if for incorrect details which isn't in the data structure has been referred to.
Whitespace entry mustn't be taken into consideration	Once the customer name input is taken in any of these methods the trailing whitespaces are removed
Exceptions mustn't throw errors and stop execution	Try-catch blocks are there in each place that might throw errors (InterruptedException for example). As for null pointers, there are empty passenger objects in areas where there aren't actual passengers.
Don't allow continuation of initial GUI without a journey and Date chosen	Editing of the Date field is disabled, and a default Date and journey have already been set to prevent this.
If a passenger has reserved many seats deleting of him should only delete a specified seat not all of his seats	Even though a passenger can book many, there are still that many passengers, so the name along with seat number is requested to delete only a passenger's specific seat
A null passenger object mustn't be considered during the process	There are checks that whether the name if each passenger is not null, if only will the execution occur (waiting room has 42 passenger objects, but might have fewer actual passengers; doing this prevents null pointers)
Upon starting the program, the correct data must be loaded into passengers	Upon starting, the details in CW-01 are loaded, accurately
Waiting room must have data accordingly to whether passenger has arrived or not	After the initial check, on whether each seat number has arrived, the waiting room is updated.

Shouldn't throw an error if R is hit before any passenger has joined the queue	If R is entered before any passenger has joined, an empty label is displayed on the GUI

EXTRAS

TEST CASES	OUTPUT
NoSQL used	NoSQL has been used as the data storage and retrieval method using MongoDB as the database
Naming conventions followed	Naming conventions have been followed. With variables having meaning, methods named as verbs and the class as a noun.
Alert boxes are validated	The Alert boxes are validated for each choice respectively. For instance, if the window closing confirmation alert returns NO the control is returned back into the GUI.
The report GUI generated must have accurate values	The values and details shown in the final report are accurate, taking into consideration everyone who was ever in the queue.