

# Group 9 Phase 1

# Contribution list

|  |  |
| --- | --- |
| Ashraf 196280 | Phase 1:   1. Responsible for implementing the **Train** class. 2. Responsible for implementing the code inside the **main** method, with the **login** and **sign up** functionality. 3. Responsible **with** my colleague **Amro** in implementing the **findUsername()** function in the Station class 4. Responsible for implementing the logic of the **seats’ price**: where the high class seats are 30egp, mid is class is 20egp, and low class is 10egp. 5. Responsible for overviewing the classes that were implemented by my colleagues; fixing any small bugs found in other classes like admin or ticket.   Phase 2:   1. Overviewed the entire group’s work, testing, and code integration. 2. Helped the group members deal with any bugs that they were facing in the GUIs/general logic of the program. 3. Implemented the following **GUI**s: Login Signup UserType ShowPassengerBookedTickets Along with their necessary code (for example, login and signup functions in Station, etc.) 4. Helped in implementing the following GUIs: HomeAdminGUI: added train button along with its code Modify\_frame: added trainId field along with its code/logic and passing this id as parameter to other forms. 5. Responsible for the **file** read and write logic: where instead of just dealing with files, we let the arrayLists’ setters and getters (in Station) update themselves based on the latest data in the .txt files (getters) and add new data to it (setters). Also, I implemented the setters and getters for **passengersList** ArrayList where my colleague Amro used these implementations in his code to update the passenger’s data. In addition, added the static counter logic, where the static counter of Person and Train changes based on the biggest id found in the respective .txt files 6. Implemented the **exceptions** classes: SignUpUserNameException TrainSeatsLimitException 7. Implemented the functions changeTrainRoute() and changeTrainTimeSlot() in Train. |
| Amro 201762 | Phase 1:   1. Responsible for implementing and testing **Passenger** class. 2. Responsible for implementing **findUserName** function. 3. Responsible for fixing some logical error in **Ticket** class. 4. Suggested to initialize arrayLists inside the constructor of class **Train**. 5. Responsible for editing functions **changeName**, **changeUserName**, **changePassword** in class **Admin**.   Phase 2:   1. Responsible for implementation and testing all the GUI forms for class Passenger except ShowPassengerBookedTickets. 2. Responsible for the user defined exception class FindTrainException. 3. Helped in writing passengerList.txt with Ashraf. 4. Responsible for changing the functions in class Passenger to fit with the GUI forms. 5. Fixed some problems related to the constructor of class Train. |
| Abdo 203795 | Phase 1:   1. Created the UML diagram  for all classes. 2. Created the class station that has three static array lists for Train,Admin and passenger and checks that username is unique in our system or not and that no one has this user name by implementing a function for that. 3. Created a class Route that has an origin, destination,distance , route price as private variables and to make sure that every distance and destination  have a different price from another,This depends on the specific destination that the user will set. 4. Created a class Ticket with my colleague Ashraf the main purpose of this class is get ticket price depending on seat position all that by implementing a function that returns the price of the seat.   Phase 2:   1. Responsible for implementation and testing all the GUI forms for class admin except Generate reports and modify account 2. Responsible for the user defined exception class Timeslot Exception. 3. Responsible for changing some  functions like add,remove and delete train in class admin to fit with the GUI forms. 4. change time slot form variable to arrayList of integers to fit with the previous feedback 5. change the class diagram to fit with the feedback and make changes on it according to new work. |
| Mahmmoud 201573 | Phase 1:   1. Drew UML inheritance between Person super class and Admin Subclass.      1. Created Admin class that allows admin to control over station’s trains by implementing functions to Add trains to the station as well as the ability to Modify or delete already created trains by passing their registered IDs in the system, also I implemented a validation logic to make sure that passed ID is already exists in our station or no, if no train associated with passed id found in our station system will raise an error message to let the admin user know that there is no train associated with passed train id, this logic is implemented on Delete train and modify train functions. There is also another validation logic I implemented on user’s input data, for variables like route distance and route price, time slot, We will check whether data which user submitted is negative or no for both variables, for time slot we added validation to make sure that time won’t exceed 24 hours based hour which we used in our system, this time slot will be used to identify when the train will go out of the station, this logic is implemented through modify and add train functions.      1. Created options in Admin class to allow logged in admin to Modify his account ‘s username and password,Account name, The functions which I used are inherited from Person class.      1. Gave admin users the ability to generate reports for specific routes by passing their associated origin and destination, before asking the user to submit the origin and destination of the route admin wants to generate report about, I gave the admin two options, first one to generate reports about Passengers numbers across this route, second one to generate report about total fare of tickets across this route. In the report which I generate for both options I collect all trains which are associated with this origin and route and then generate a report with trains and numbers of passengers as well as number of tickets according to their categories (High,Mid,Low) class, after that I generate report with total fare for all tickets categories, and multiply them with their constance price saved in our system.   Phase 2:   1. Created GUI for admin ‘s home where admin can go through other forms easily, forms that gives him the ability to Generate reports, Manage trains, Add Trains, Manage his account 2. Created a Generate report form where user can submit origin and destination for route that sh/he wants to generate a report for and created a user defined exception to alert him that there is no train in submitted route, after that admin choose which type of reports sh/he is looking for, whether Passenger or total fare. 3. Created Report generation GUI that represents data in a table that is suitable for the admin to get more insights about stats 4. Created Manage account GUI form and implemented an exception which is created by my colleague Ashraf to check whether username already taken or no and show error message accordingly 5. Updated some Admin class functions from previous phase to stop receiving inputs through Console and implemented them in GUI (Generate Report, Modify account) 6. Helped in Modifying add train GUI. 7. Implemented getAdminList() and setAdminList() functions in Station class with Java files exceptions. |

# UML Diagram