**Railway Management System**

**CSE 4301 – Final Project**

**Blueprint**

Please go through the project proposal to get an overview of the program before reading this document.

**Essential Components**

The main parts of the program essentially consist of a Station class, a Route class and a Train class.

The Station class will contain the location of that station and will contain information regarding the stations to which it is connected. Whoever chooses to work on this can handle this problem in any way they wish, but I would recommend using an array of pointers.

The Route class will contain a route number, arrays that point to different Station objects, and another array that contains the time at which any particular train will be at that station. This means that if one train is at station A at time t1 and station B at time t2, and another train is at station B at time t1 and station A at time t1, then those are two different routes. Essentially, every train must be following a different route, since two trains cannot be on the same route at the exact same time. It would be best if it was possible to store the time information in the format of actual time and not just an integer. Route information should also ensure that no stations are skipped over, i.e. if station C is between station A and station B and there is no direct connection between stations A and B, then the route should not directly connect station A to station B, but should go through station C.

The Train class will contain an id for the train, the route which the train follows and the number of seats in the train.

The actual visible functionality of the program are separate functions that make use of the classes outlined above. It would be best to include all of these functions in a separate header file, in order to keep the main file clean and short. When a user tries to book a ticket, they will be asked for a departure and arrival location. Given this information, the Route class will be used to present possible times. The Route class must therefore accommodate functions that allow this. Once the user has selected departure and arrival times, the specific routes that match with those times will be selected, and trains that use those routes will be presented to the user. The Route and Train classes will have to work together to achieve this. There could be difficulty keeping track of the trains here. A solution needs to be discussed when the problem arises. Once the user has made a choice of train, the number of available seats on that train should be decremented. Furthermore, trains that do not have any available seats should not be presented in the first place. It is recommended that the seat numbers of all trains be kept to a minimum at first for testing purposes. Once everything is complete, a ‘ticket’ should be presented to the user.

It is essential to store information regarding stations, trains and routes in some manner since it would be cumbersome to repeatedly input the information every time the program is run. For the time being, this could be done using .txt files, but this is a very bad idea and we need to think of some way to some way to scramble the data and make it inaccessible from outside the program.

**Additional Functionalities**

There is a large number of additional functionalities that can be added to the program, but it is highly discouraged to work on any of these before finishing the main functionality outlined above.

An administrative view is planned that allows information regarding stations, trains and routes to be edited. This could be troublesome for two reasons. Firstly, a lot of information needs to be checked if anything is edited since there is a good possibility that the program might break otherwise. Secondly, an administrative view would require a login page, which could be extremely difficult to properly implement. Further, we would need to keep track of administrators and their login information.

A timetable could be made that shows the schedule for different trains, including which stations they stop at and at what times. The Route class should come in handy here.

The Route class itself makes the assumption that there will be absolutely no delays. It would be interesting if we could edit information in the Route class randomly as the program is running, simulating delays due to different reasons.

Stations in the current program are assumed to be able to contain any number of trains. This is impractical. The program could include a check to ensure that the maximum number of trains that a station can hold is not exceeded at any given time.

The Train class could be extended to include different categories of seats i.e. Economy, Business etc. and seat numbers adjusted and checked accordingly.

It would be useful to have a customer profile of sorts, that keeps track of the tickets that a certain customer has purchased. This would of course, require the use of a new Ticket class to hold all that information. Creating a customer profile would again present the login problems faced with the administrator view.

Finally, and most importantly, a graphical interface should be made to accommodate all possible functionality. This is the most important additional feature and someone should begin working on it as soon as the basic requirements have been fulfilled.

**General Information**

An Excel sheet that contains the different parts of the program, their priority and who is currently working on them has been attached. No one has been assigned to any particular parts so feel free to pick any of the items that are unassigned and begin working on them.

It is requested that you not try to simply add files into the project folder. This causes problems since CodeBlocks does not automatically detect them. Please create any classes and related files through the CodeBlocks IDE. The basic project has been set up already. Please use the .cbp file to open the project. A shortcut to the .cbp file has been kept in the main directory as well for ease of access.

Please refrain from working on any files that someone else is already working on. Updating the same file together causes problems in GitHub regarding which version to keep. This should be avoided if at all possible. If it is essential that two or more people work on the same file at the same time, please communicate with each other before touching those files in order to avoid the errors.

When committing updates to GitHub, please try to give a sensible description to what work has been completed.

Please feel free to ask any questions, and suggest any new ideas in the Messenger group. Please clear up any confusion you have immediately, and before beginning your work.