CSC 336 Spring 2018 Railroad Project Version 1

The project consists of two parts.

(1) All teams will create a basic railroad between Washington and Boston with trains, stations, train schedules, passengers and a simple reservation system. You can call it whatever you want, just not Amtrak.We have thoroughly discussed the data modeling for the railroad in class, and used it on the mid-term exam.

We will go back to using MariaDB on the "slow" server for lab work because the Web Server and programming tools are installed there.

To save time, I have given your teams select access to two versions of the railroad database developed by students last spring. They are called "railroad1" and "railroad3" and are similar but not identical. Look for them on MariaDB10. You can use these pretty much "as is" for the basic reservation system GUI.

(2) Each team will choose one of 7 enhancements to the railroad such as extending the line north and south, implementing differential fare schedules, implementing additional express trains, implementing high speed trains, implementing a train status and notification of lateness.**If you use the sample schemas and data in "railroad1" or "railroad3" you will need to add either data, or additional tables, or both.**

I suggest that you divide the labor among your team members along these lines:

(1) Creating and testing the database manipulation routines
(2) Creating and testing the GUI
(3) Creating users, passengers, trips and reservations; filling up trains
(4) Managing the project, writing reports, managing repositories.

Your progress report and final submission must specify what each team member contributed, which will likely be to more than one of the 4 divisions.

Your suggestions for improvements are welcome, and please put them in writing so I can keep track of them.

The milestones for the Railroad Project are

(1) Each team chooses a platform and language in which to write the reservation GUI, and an enhancement to implement. The choice of language and platform needs to be discussed with me; the choice of enhancement is “first come, first serve”. **Choices are due May 4**.

(2) RR Project Base Reservation System code with an ER diagram and flowchart+ enhancement is due on **Thursday May 17**.

Requirements of the basic railroad:

(1) Implement data structures including tables and high level language constructs to model
   a. Trains
   b. Stations
   c. Stops that each train makes with arrival and departure times
   d. A means of recording what segments of the route each passenger books
   e. A record of how many seats are sold (or are free) for each segment of the route
   f. A record of passenger reservations up to a year in advance.

(2) Parameters and simplifications
   a. Washington and Boston are termini; include all intermediate stops as on the actual Northeast Regional. Use reasonable time intervals between stops.
   b. Each train has 7 passenger cars with 64 seats in each.
   c. Calculate the price per segment based on time or mileage between stations
   d. Create 8 trains per day in each direction (northbound and southbound), 3 morning, 3 afternoon, 2 evening. Use different train numbers for each train, and different schedules/numbers on weekends.
   e. For the basic reservation system:
      i. Assume that the trip is continuous with no gaps or stopovers
      ii. Assume only one fare class
      iii. Assume booking for one person only
      iv. Assume cancel and rebook, not modified reservation
      v. For each calendar year, publish one schedule 6 months in advance.

vi. Delete or otherwise mark off completed train/dates and passenger trips.

f. The Reservation GUI and high level programming can be implemented in a variety of languages and platforms. You can use PHP, node.js ,Python CGI or Java Swing with Xwindows in the lab, or you can use Python with Django or PHP or node.js on the Google Cloud Platform. The result must be available on a publicly accessible web site, whether in the lab or on GCP. It is not acceptable to present the result on your laptop, or to expect me to install products on my laptop or office system. It is also not acceptable to use SQLLite or another not-fully-functional DBMS.
    i. The Reservation GUI should display available trains for the passenger's itinerary and choice of date and morning, afternoon or evening. That is, if some portion of a trip from New York to New Haven is booked up, then that is unavailable and other trains should be displayed. From the displayed trains, the user should make a choice, then be directed to a reservation screen.
    ii. In the basic railroad, there is no assigned seating. Just book the whole train until it fills up (i.e keep track of free seats for each segment). Of course, some segments may fill up before others. Be sure to simulate the condition in which segments are full.
    iii. The Reservation page of the GUI should record a passenger's name and some payment method, providing a unique passenger ID and a unique ticket number.
    iv. The Reservation GUI should be able to display a passenger's reservation and allow for cancellation and rebook. Do not bother with penalties or surcharges for the basic system.

g. Concurrency control. You will need to protect the routine that makes the actual reservations from interleaving. You can do this with a mutex in a high level language or with an exclusive lock or a transaction isolation level in MySQL. I will help you with either approach. The reason for some kind of concurrency control is that you do not want an available trip to disappear between the time the user chooses it and the time the reservation is actually made – two separate passes through "seats_free".

Enhancements (each team chooses one):

(1) Generate passenger volume that fills up popular itineraries and times of day. Use some randomization and batch inserts to do this. Based on demand, **create new trains** to meet the demand. These trains will not make low volume intermediate stops. They are not high speed trains, but skip-stop or expresses. You can charge a higher flat rate for these trains. You must not crash them into the slower trains.

(2) Implement assigned seating for individuals, couples and larger groups. Allow a choice of window or aisle seat. Number the seats airline style to allow you to compute placing couples and groups as close together as possible. You do not have to give a choice of car, just put the couples and groups wherever they fit, and fill in singles with strangers.

(3) Implement differential fares. Increase the fares as the departure date approaches. Give discounts for seniors, children and the military. Charge surcharges for pets. Optionally, reserve one car for Business Class.

(4) Implement “train status” on the web site so passengers can see when a train is due to arrive at a station. Make your trains late using randomization, and optionally make them late enough to delay other trains. Implement email alerts for reserved passengers. Yes, you can send email out of the lab to CUNY email addresses.

(5) Extend your railroad south to Virginia and north to Boston and Burlington, Vt. The real Amtrak has branches of the Northeast Regional that originate in and return to either Lynchburg, VA, Roanoke, VA or Richmond, VA, and there is connecting service north to Vermont and Maine. You can “imagine” that there is no change of stations to get to Maine.

(6) Modify the GUI to allow existing reservations to be changed, rather than cancelled and rebooked. Allow a change in time of day, itinerary, date, assuming seats are available. Note that this may change the total fare in several ways – more expensive train, less expensive train, etc. Charge a fee.

(7) Implement high speed trains, two in the morning, two in the afternoon, and two in the evening in either direction. Run them 50% faster than the locals with stops only at major stations with Boston and Washington as termini. For simplicity, allow them to pass the locals only in stations. Figure out an algorithm to schedule them so that they don't get stuck behind the locals. Include them in the reservation GUI with higher fares – all Business Class.