CS 202 Project

Railway Management System

Due Date: 07/01/2022

TAs: emir.arditi@ozu.edu.tr, cihan.eran@ozu.edu.tr

This assignment covers querying a SQL database using Java JDBC. When writing your program, you should create a new schema/database in your MySQL server just for this assignment. You must implement your database design by considering the functional dependencies and your database should be 3NF. Keep in mind that since this is a database course, it is expected that you solve as many challenges as you can using SQL. Hacky-Java Based solutions will result in point deduction!

In this project you are expected to implement the entities and endpoints for a server back-end Java application initialized with the Spring framework. You can consider the system to be implemented as a backend for a Railway Management System website. Some features include adding/removing trains, creating schedules, etc. Base code, sample entities, sample endpoints and sample tests are all provided.

The framework setup requires you to write your DDL SQL statements into **"src/main/resources/db-schema.sql"** file. And for each table you create, you have to also create a Java class that shares the same name with the table you have created, under the "**src/main/java/edu/ozyegin/cs/entity"** folder. An example of it for the "Sample" table is present in the code base. Controllers that contain endpoint handlers, should be under the "**src/main/java/edu/ozyegin/cs/controller**" folder.

You are not to touch and/or modify the files under the "src/test" folder. During evaluation, a different "test" folder will be used instead. Modifications you make in this folder will be discarded. Under no circumstances should you be required to modify other components of the project. If you encounter an issue with the contents of the project, *contact TAs*.

You are responsible for designing a Railway Management System. This system contains stops, routes, trains and schedules. Each of these components have unique semantics and these semantics are described below:

1) Stops
    a) Each stop must have a unique name
2) Routes
    a) Each route must have a unique name
    b) Each route can contain arbitrary amounts of stops

        c) The order of the stops is not important
3) Trains
    a) Each train must have a unique name
4) Schedules
    a) Schedules must contain a route, a main train and time
        i) Ex: (Route: DT2, Train: Barbaros Train, Time: 13:30)
    b) Schedules can also have an arbitrary amount of backup trains if something happens to the main train

Endpoints

When an API interacts with another system, the touchpoints of this communication are considered endpoints. For APIs, an endpoint can include a URL of a server or service. Each endpoint is the location from which APIs can access the resources they need to carry out their function.
Following table lists the API endpoints that you are expected to implement. **Input** column defines the fields that will be sent to the endpoint. **Output** defines the expected output of the endpoint. Examples regarding the way these are handled are present in the sample code provided. If you have any questions, please contact TAs.

Endpoint	**Input**	**Output**
`/stop/create` **POST** Create a Stop	StopName: String	success: boolean - "true" if successful, "false" otherwise
`/stop/modify/rename` **POST** Rename a Stop	StopId: int NewName: String	success: boolean - "true" if successful, "false" otherwise
`/stop/modify/delete` **POST** Delete a Stop	StopId: int	success: boolean - "true" if successful, "false" otherwise
`/stop/get_all` **GET** Get All Stops	-	stops: JSON Object Array - An array of Stops. Each object in the array is a Stop data from the table, and all of its data.

`/route/create` **POST** Create a Route	RouteName: String stop_ids: int[] (int array)	success: boolean - “true” if successful, “false” otherwise
`/route/modify/rename` **POST** Rename a Route	RouteId: int NewName: String	success: boolean - “true” if successful, “false” otherwise
`/route/modify/add_stop` **POST** Add Stop to Route	RouteId: int StopId: int	success: boolean - “true” if successful, “false” otherwise
`/route/modify/remove_stop` **POST** Remove Stop From Route	RouteId: int StopId: int	success: boolean - “true” if successful, “false” otherwise
`/route/delete` **POST** Delete a Route	RouteId: int	success: boolean - “true” if successful, “false” otherwise
`/route/get` **POST** Get Info of Route	RouteId: int	route: JSON Object - A JSON Object that contains all the information about a route. - **stops**: JSON Object Array - List of stops that are in the route. - Each object in the array is a Stop data from the table, and all of its data.
`/route/get_all` **GET** Get All Routes, Including their stop info	-	routes: JSON Object Array - An array of Routes. Each object in the array is a Route data from the table. - **stops**: JSON Object Array - List of stops that are in the route. - Each object in the array is a Stop data from the table, and all of its data.
`/train/create` **POST** Create a Train	TrainName: String	success: boolean - “true” if successful, “false” otherwise
`/train/modify/rename` **POST** Rename a Train	TrainId: int NewName: String	success: boolean - “true” if successful, “false” otherwise
`/train/modify/delete` **POST** Delete a Train	TrainId: int	success: boolean - “true” if successful, “false” otherwise

`/train/get_all` **GET** Get all Trains	-	trains: JSON Object Array - An array of Trains. Each object in the array is a Train
`/schedule/create` **POST** Create a Schedule	RouteId: int TrainId: int Time: String (unix timestamp)	success: boolean - “true” if successful, “false” otherwise
`/schedule/modify/change_time` **POST** Change time of a Schedule	ScheduleId: int Time: String (unix timestamp)	success: boolean - “true” if successful, “false” otherwise
`/schedule/modify/change_route` **POST** Change Route of a Schedule	ScheduleId: int RouteId: int	success: boolean - “true” if successful, “false” otherwise
`/schedule/modify/change_train` **POST** Change the Train Used in the Schedule	ScheduleId: int TrainId: int	success: boolean - “true” if successful, “false” otherwise
`/schedule/modify/add_backup_train` **POST** Add a backup train to schedule	ScheduleId: int TrainId: int	success: boolean - “true” if successful, “false” otherwise
`/schedule/modify/remove_backup_train` **POST** Remove a backup train from a schedule	ScheduleId: int TrainId: int	success: boolean - “true” if successful, “false” otherwise
`/schedule/delete` **POST** Delete a Schedule	ScheduleId: int	success: boolean - “true” if successful, “false” otherwise
`/schedule/get_all` **GET** Get All Schedules	-	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/route/stop` **POST** Get all routes that pass from the given stop	StopId: int	Ids: JSON Object Array - An array that contains RouteIds only.
`/statistics/route/size` **POST** Get all routes that have at least K stops	K: int	Ids: JSON Object Array - An array that contains RouteIds only.

`/statistics/stop/size` **POST** Get all stops that are included in at least K routes	K: int	Ids: JSON Object Array - An array that contains StopIds only.
`/statistics/stop/time` **POST** Get all stops that are present on route which starts in a given Time	Time: String (unix timestamp)	Ids: JSON Object Array - An array that contains StopIds only.
`/statistics/schedule/stop` **POST** Get all schedules that pass from the given stop	StopId: int	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/schedules/time` **POST** Get all schedules that start at the given time	Time: String (unix timestamp)	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/schedules/route` **POST** Get all schedules that uses the given route	RouteId: int	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/schedules/time_stop` **POST** Get all schedules that start at the given time and pass from the given stop	StopId: int Time: String (unix timestamp)	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/schedules/backup_count` **POST** Get all schedules that have at least K backup trains	K: int	schedules: JSON Object Array - Contains all schedule data available in the database, and their contents.
`/statistics/train/backups` **POST** Get all trains that are backup trains on at least K schedules	K: int	Ids: JSON Object Array - An array that contains all available Tren data
`/statistics/train/max_main` **GET** Get the train which is listed as the main train on most schedules	-	Id: int - An array that contains Ids only.
`/statistics/train/stop_backup` **POST** Get all trains that are listed as backups on schedules which passes from the given stop in route	StopId: int	Ids: JSON Object Array - An array that contains TrainId only.

Prerequisites

- Install Docker (https://www.docker.com/get-started)

Install JDK 14 . This project requires you to use JDK 14 with it.

Using Codebase

After installing prerequisites, you can test the output of your code by creating test cases and running them.
Sample test cases are present in the codebase provided to you.

Submission Guideline:

1) Please compress all your files and submit a single *zip* file.
2) The name of your zip file should look like the following: 'mohammed-lee-S000123-HW1.zip'
3) Do not change the structure of your project aside from modifying and/or adding files as mentioned above.
4) Your Database design must comply with 3NF.
5) Write a report detailing your design decisions.
6) Your report **MUST** also include an ER diagram representing your database design, your functional dependencies and description of the design decisions you made and why you have made them.
7) Your JAVA codes should be in the same folder structure as the way they are in your computer. **DO NOT CHANGE THEIR LAYOUT. DO NOT TAKE YOUR FILES OUT OF THE FOLDERS THEY ARE IN.**
8) Failing to comply with these guidelines will result in a **penalty**.
9) When extracted, your code should be able to run without an error. Your project should be able to open properly.

For any questions, please contact TAs