CST2550 Coursework 1

Dr Barry D. Nichols

December 17, 2019

1 Brief Task Description

Design and implement a database to store personal trainer bookings for a gym and a client/server software system in Java to interact with the database. The client program should have an intuitive user interface to add, remove and manage bookings. The server program should allow multiple clients to connect at the same time and interact with the database (see the scenario below).

2 Submission

You must submit a single **zip** file of all required source code and a single **PDF** file of your report by **Friday, 31$^{\text{st}}$ January 2020**.

The source code zip file should include:

- SQL to create and populate a MySQL database
- Java source code for the client and server program(s) to interact with the database

Note:

- **Only source code should be included in the zip file, not IDE files**
- **Code must compile and run on the Linux lab setup, code which does not compile on the lab setup will achieve a *maximum* of 40%**
- **As anonymous marking will be applied, you should *not* include your name in your source code or report**

3 Scenario

A gym has several personal trainers (PTs) who each have one or more specialisms. Currently PT bookings are managed using a paper form and a diary. When a client books a PT session the staff member completes the form which is then filed in the filing cabinet and also adds the booking to the gym diary. The form includes the fields:

- client name
- personal trainer name
- date
- time
- duration
- focus (e.g. weight loss, muscle gain, flexibility)

This system works but as the gym is becoming more popular it is becoming time consuming. Particularly when multiple members of staff want to take, check and/or update bookings at the same time.

Computers are located at locations throughout the gym and any staff member will be able to use it to check, update or create a new booking.

Allowing clients to view their own bookings (e.g. on the website) could be possible in the future, but is outside the scope of the current project, which will only be accessed by staff.

4 Detailed Software Description

It is recommended that you complete the tasks in the following order as the later sub-tasks will require the earlier ones.

4.1 Database

Design a normalised (3NF) relational database for the above scenario, and create an ER diagram of the design. Ensure the multiplicities, domains and keys are specified correctly.

Write the SQL to create the tables and insert some sample data.

Once you have the database designed and implemented, design the SQL queries required to allow the required functionality, i.e.:

- add booking
- update booking
- delete booking
- list bookings
  - for given PT
  - for given client
  - for specified date

4.2 Server

Design and implement a (command line) server program which connects to the, previously implemented, database. It should have methods to achieve the required functionality (using the SQL queries you developed for the database):

- Add booking
  - shouldn't allow double bookings
- List bookings
  - all bookings
  - bookings for a given PT
  - bookings for a given client
  - bookings on a particular day
- Update booking
  - shouldn't allow double bookings
- Delete booking

It should also have a method to load the given data file and insert it into the database.

The sever program should provide the application protocol in Table 1 to clients using TCP streams (the server can be tested using Telnet).

Add the required code to allow multiple client programs to connect to the server simultaneously.

Table 1: Application Protocol

Command	**Description**	**Server Response**
ADD *booking-details*	adds the booking to the database (if no clashes)	confirmation of success or failure (with reason)
LISTALL	Retrieve a list of all bookings in the system	Returns a string of all bookings (separated by newlines)
LISTPT *pt-id*	Retrieve a list of all bookings in the system for the given PT	Returns a string of all bookings of the given PT (separated by newlines)
LISTCLIENT *client-id*	Retrieve a list of all bookings in the system for the given client	Returns a string of all bookings of the given client (separated by newlines)
LISTDAY *date*	Retrieve a list of all bookings in the system on the given date	Returns a string of all bookings on the given date (separated by newlines)
UPDATE *booking-id booking-details*	Update the booking with the specified id	Confirmation of success or failure (with reason)
DELETE *booking-id*	Remove the booking with the specified id from the system	Confirmation of success or failure (with reason)

4.3 Client

Design and implement Java class(es) to be used by the client GUI to connect to the server and (using the application protocol) execute the required functionality (this can be tested with a basic command line program).

Design and implement an intuitive JavaFX GUI program which utilises these Java class(es). This GUI program should allow the staff to add, list, update and delete PT bookings. It should also have a method to disconnect from the server (freeing up the server for other clients).

5 Report Layout

The sections should be as follows:

- Abstract
  - a paragraph describing the work, results and conclusions
- Introduction
  - brief description of *your* work **(not the coursework specification)**
  - brief (paragraph) description of the layout of the rest of the report
- Database design
  - written description
  - ER diagram
  - normalization
- Software design
  - written description
  - UML diagrams
  - GUI wireframes
- Testing
  - description of testing approach used
  - record, e.g. tables of tests completed
- Conclusion

  - summary of work done
  - limitations of the current design/implementation and critical reflection of your work
  - how you would approach a similar project differently in future

- References
  - any references here should have matching in-text references
  - any work which is not your own must be referenced
  - using the Harvard reference system

6 Academic Misconduct

This is individual work and you should complete it yourself. You should not work as a group and each submit the same work (even with minor changes) as your own. Any material or ideas found online, in textbooks, etc should be properly referenced.

You should familiarise yourself with the university's academic integrity and misconduct policy:

https://www.mdx.ac.uk/about-us/policies/university-regulations

7 Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet an assessment deadline or affect your performance in an assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse and outside of your control. Please see link for further information and guidelines:

https://unihub.mdx.ac.uk/your-study/assessment-and-regulations/extenuating-circumstances

8 Marking

The report and included code will be marked according the to attached marking scheme. Marking will be **anonymous**, i.e. the marker will not see the name of the student while marking.

9 Feedback

Provisional marks and written feedback will be available on Moodle within 15 working days of your submission. If you would like clarification or more detailed feedback on your coursework contact your module tutor.

10 Marking Scheme

10.1 Report

Item	**Marks**
Abstract	5
Introduction	8
Database design (correct description, ER diagram and normalisation (3NF)	8
Software design (description, UML diagrams and GUI mock-ups/wire-frame diagrams)	8
Software testing (description of testing approach used and evidence of testing)	8
Conclusion, including: - summary of work done - limitations and critical reflection - how would change approach on similar task in future	8
Layout and clarity or writing	5

10.2 Code

Item	**Marks**
Code quality (e.g. comments, layout, exception handling, input validation, etc.)	10
Database implementation (matches design)	5
Implementation of specified application protocol in client and server.	5
Client/server application using TCP for communication	10
Client user interface (intuitive, GUI)	10
Implementation of required functionality	10