# CS2102 Database Systems

Semester 1 2019/2020

Project

**DELIVERABLES: 9 November 2019** **20 MARK**

The objective of this team project is for you to apply what you have learned in class to design and develop a web-based database application. The project is to be done in teams of four students.

## 1 Application Topics

Each project team will be *randomly* assigned to one of the five application topics. The following provides a brief description of each application topic and some example concrete application deployments. As the topic descriptions are not meant to be complete specifications of the application topics, each team has the freedom to decide on the application's data requirement as well as the application's functionalities/features to be implemented. Your database design and implementation for the application should demonstrate non-trivial usage of SQL and database features.

### Topic A: Crowd Funding

This is a crowd funding system (e.g., https://www.kickstarter.com/ or https://www.indiegogo.com/) to allow an aspiring creator to get funding for their project. Projects can be almost anything such as movie productions, board games, or electronic products. Users of this application are either creator looking to fund their projects or funder funding parts of a project. The application provides templates for generic common project to facilitate creator to create new projects. The project must have a deadline and it is funded if it meets or exceeds the funding requirement by the deadline period. Funder may cancel their funding before the deadline but not after. Creator may also fund other creators' projects. Each user must have an account.

### Topic B: Course Registration System

This is a course registration system (e.g., http://www.nus.edu.sg/ModReg/) that allows user to register for their preferred course. University administrators can create a course for students to register. Students are required to register for a course before a deadline. To register for a course, the student must satisfy the course pre-requisite which is set by administrator. Each course should have a time period and a quota where the quota is applicable for the given registration period. In particular, if the number of students registering for the module exceed the quota, students are accepted automatically into a course according to some criteria. However, the administrator may manually accept students and override the quota. Each user must have an account.

### Topic C: Course Management System

This is a course management system (e.g., https://luminus.nus.edu.sg/) that allows professors to manage a course he/she is teaching. Students may request to enter a course. Professors may categorize students in the course into groups such as the different tutorial groups. Professors may create forums where only certain groups or collection of groups in the course can read or write. However, the professor may read, write, or delete entries in the forum. Additionally, professors may add teaching assistants to manage groups. Teaching assistants cannot create groups or forums but can read, write, or delete entries in the forum. Each user must have an account.

### Topic D: Car Pooling

This is a carpooling application that allows car drivers to advertise opportunities for carpooling and passengers to search for car rides (e.g., https://www.sharetransport.sg/). A car ride advertisement specifies (among other information) an origin to a destination on a certain date and a given time. Both users and cars have a profile. Drivers advertise rides and passengers bid for the rides. Users can play both roles of drivers and passengers. The successful bidder for an advertised ride could either be chosen by the car driver or automatically selected by the system based on some criteria. Each user must have an account.

**Topic E: Restaurant Reservation**

This is a restaurant reservation application that allows diners to book reservations at restaurants (e.g., https://www.chope.co/). Restaurants can advertise their availability (e.g., cuisine type, branch locations, opening hours, menu prices, etc) and diners can search for restaurants to book reservations by providing various information (e.g., date and time, cuisine type, number of people, budget, preferred locations, etc) and rate restaurants based on their dining experience. Each booking can be confirmed based on various criteria (e.g., booking time, availability, number of diners, etc). Diners could cancel or make changes to their reservations. The system could provide various incentives (e.g., award points for each confirmed booking) to retain users. Each user must have an account.

## 2 Application Requirements & Functionalities

The data models of your application must satisfy the following requirements:

- The total number of entity sets and relationship sets must be at least 15.
- There must be at least one weak entity set.
- There must be at least three non-trivial application constraints that cannot be enforced using column/table constraints and must be enforced using triggers (e.g., multi-table constraints).
- If there is a suitable candidate key(s), you are NOT allowed to use serial type[1]. Every serial type must be well-justified to be used.

Each application must provide at least the following functionalities:

- Support the creation/deletion/update of data.
- Support the browsing and searching of data.
- Support at least three interesting complex queries on the data.
  - An example of an interesting query is one that performs some data analysis that provides some insights about your application.
  - A simple `SELECT-FROM-WHERE` cannot be considered a complex queries.
  - The complex queries must contain at most two `CTE` (common table expressions).

Your application should not be limited by the functionalities of its brief description given in the previous section (Section 1). You are free to introduce interesting functionalities to make your non-trivial (e.g., requiring complex queries, transactions, triggers, etc). You are also free to use any of PostgreSQL's features and other SQL constructs beyond what are covered in class.

For the final project demo, your application's database should be loaded with reasonably large tables. To generate data for your application, you can use some online data generators (e.g., https://mockaroo.com/ or https://www.generatedata.com/). However, we would suggest you to *write your own program to generate the SQL insert code*.

## 3 Software Tools

The architecture of your application should follow a typical three-tier (clients, application server, and database server) architecture of a web-based database application shown in Figure 1 with web-browsers as the clients and a web-server as the application server.

To get started, you can download the following files:

- `PostgreSQL Installation Guide.pdf` from LumiNUS's SQL folder
- `NodeJS Start Guide.pdf` from LumiNUS's Project folder
  - If you wish to use `PostgreSQL-ExpressJS-NodeJS` stack
- `Python Flask Start Guide.pdf` from LumiNUS's Project folder
  - If you wish to use `PostgreSQL-Flask` stack

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

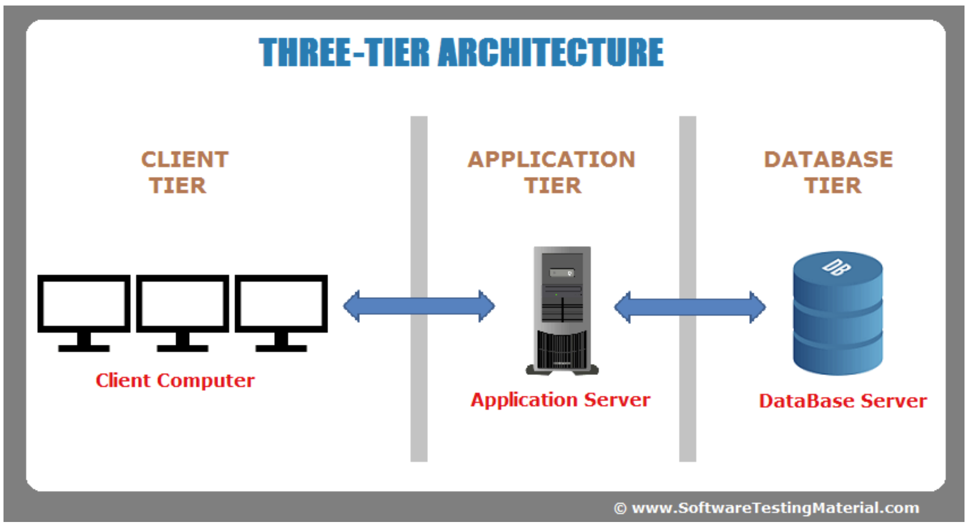

Figure 1: Three-tier web-based application architecture.

You are also welcomed to use other frontend/backend development stacks for your web application such as Angular JS[2], React JS[3], Laravel[4], Symfony[5], or Apache[6].

## 4 Project Deliverables

The project has two deliverables which are due on 9 November 2019 by 11:59pm.

- **Project report** (*up to a maximum of 20 pages in PDF format with at least 10-point font size*). The report should include the following.
    1. Names and student numbers of all team members (that start with **A** instead of **E**), project team number, and project topic (*on the first page*).
    2. A listing of the project responsibilities of each team member.
    3. A description of your application's requirements and functionalities.
        - Highlight any interesting/non-trivial aspects of your application's functionalities/implementation.
        - Describe and explain all constraints in your application.
    4. The ER model of your application (which must satisfy the requirements in Section 2). If your ER model is too large (spanning more than a page), you may want to either (a) include a single-page simplified ER model with non-key attributes omitted, or (b) explain your ER model piece-wise such as for every relation set and entity sets connected to the relation set. You can then present the more detailed ER model.
    5. The relational schema derived from your ER data model; i.e., show the DDL statements of all your database tables.
        - If there is any table that is not in 3NF/BCNF, justify your design decision.
        - If all your tables are in 3NF/BCNF, briefly explain.
    6. An English description of each of the three non-trivial constraints that are enforced using triggers. Show the code of your trigger implementation.
    7. Specification of the software tools/frameworks used in your project.
    8. Two or three representative screenshots of your application in action.

[2] https://angularjs.org/
[3] https://reactjs.org/
[4] https://laravel.com/
[5] https://symfony.com/
[6] https://www.apache.org/

9. A summary of any difficulties encountered and lessons learned from the project.

- **Project source code** including a `README` file describing how to deploy and run your application.

We would suggest deploying your web-based database application online such as on Heroku[7] which support `PostgreSQL-ExpressJS-NodeJS` stack.

## 5 Project Deadlines

The following table lists the project deadlines.

| Due Date | Project Task |
|---|---|
| Week 8 (15 October 2019) | 5-min Alpha Demo (during tutorials) |
| Week 12 (9 November 2019) | Report & Code Submission |
| Week 13 (12 November 2019) | 15-min Final Demo (during tutorials & evenings) |

**Alpha Demo (Week 8).** Each team is to present a 5-minute demonstration to show the progress made in the project. The alpha demonstration will take place during Week 8's tutorial hours. Each team is to show the constraints identified for the application as well as the ER data model to capture the constraints. Additionally, if there are any working functionalities, you can also show them during the demonstration. The booking of the alpha demo time slots will open during Week 6.

**Final Demo (Week 13).** Each team will present a 15-minute demonstration of their project showing the main features and answering questions from the evaluators. The final demonstration will take place during Week 13's tutorial hours and in the evenings. The booking of the final demo time slots will open during Week 11.

## 6 Evaluation Criteria

The maximum score for the project is 20 marks with the following breakdown:

- **Alpha Demonstration:** 1 mark
- **Report and Final Demonstration:** 19 marks
  - Database design (*6 marks*)
    - ER data model (*3 marks*)
    - Relational schema (*3 marks*)
  - Interestingness/complexity of application (*6 marks*)
    - Interesting complex queries (*3 marks*)
    - Non-trivial constraints enforced with triggers (*3 marks*)
  - Application design, functionalities, and interface (*3 marks*)
  - Report organization and presentation (*2 marks*)
  - Final demo presentation (*2 marks*)

The top-10 projects will each receive 1 additional mark (if the total marks is below 20).

## 7 How to Submit

- Submit your source code by creating a zip file named `codeNN.zip`, where `NN` is your project team number. Upload this file into the LumiNUS folder named `Project-Code-Submission`.
- Submit your report by creating a PDF file named `reportNN.pdf`, where `NN` is your project team number. Upload this file into the LumiNUS folder name `Project-Report-Submission`.

The deliverables are due on 9 November 2019 by 11:59pm.

[7] https://www.heroku.com/

For late submissions, submit to the LumiNUS folder `Late-Project-Code-Submission` and `Late-Project-Report-Submission`. One mark will be deducted for each late day up to two late days; projects submitted after the second late day will receive zero marks and will not be graded.