Mini web application

Mini web application which is created to provide an opportunity for candidates(users) who are applying for exams in available testing centers.

Database design

The database (DB) based on MySQL server.

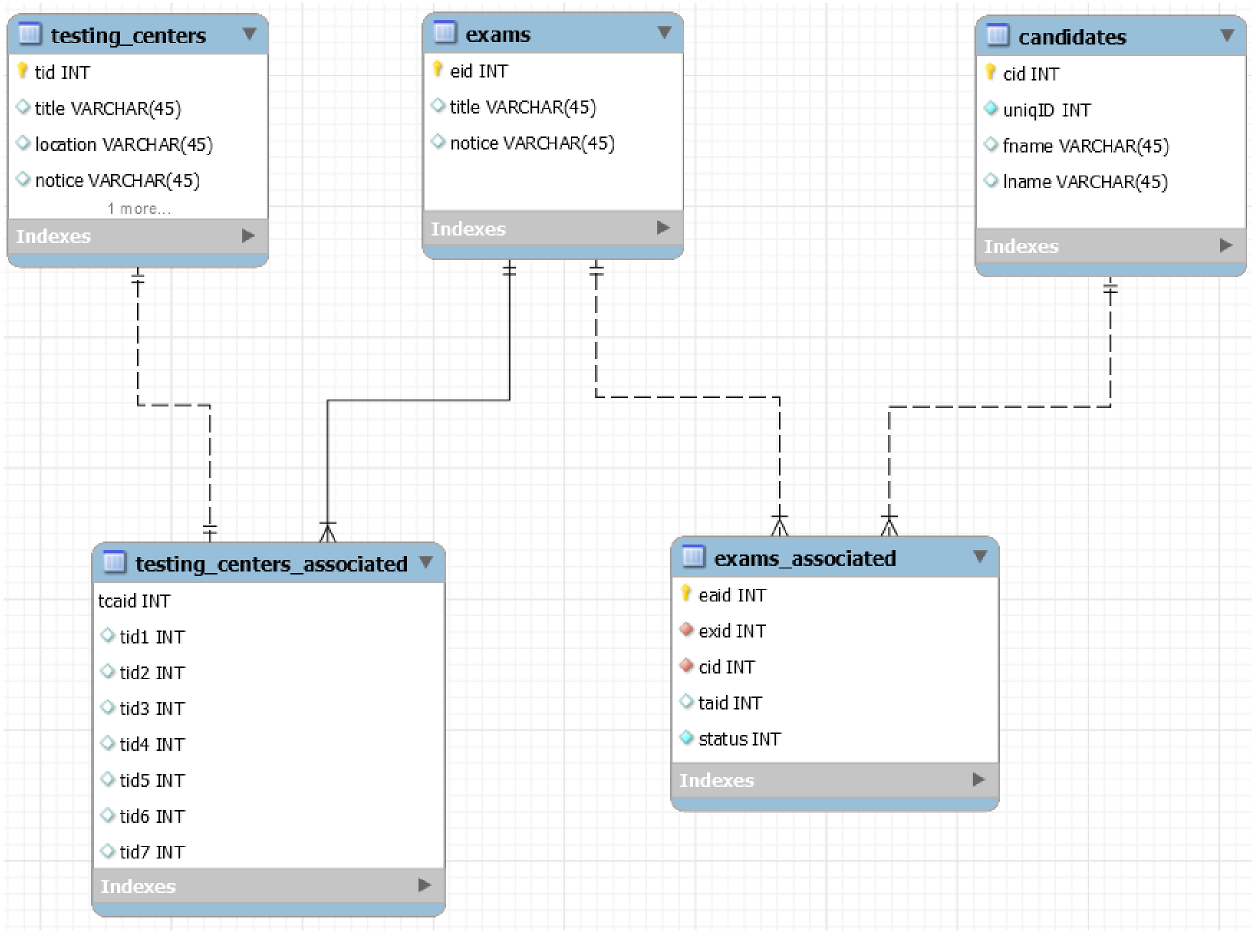


Fig. 1 – MySQL database design

The DB designed using by LinkedList data structure pattern. It provides an opportunity to extract data efficiently across all DB tables. For example, using only userID (uniqID in Fig. 1) the DB provides an access to the 'candidate' table then by extracted 'cid'-key for the 'exam_associated' table then to the exams table and so on. Basically, description of the each of the tables are presented below in the Table 1.

Table 1. Database table description

Table name	**Description**
Exams	Contains exam information like title, notice and internal ID.
testing_centers	Contains testing center information like title, location, notice and internal ID.
Candidates	Contains information about users like unique userID, frist/last name and internal ID
testing_centers_associated	Contains information about available testing centers for each exam i.e. each exam could be support up to 7 testing centers
exams_associated	Contains information about associated exams with particular candidate like examID testing_centerID and status

Application design

The application is based on the next components:

Application.cfc – basic main backend class which provide access to the life cycle events like 'onRequestStart()' when current page is processed in the browser.

Index.cfm – basic main frontend class which provide access to the page handling events like text entering/button picking and rendering.

db_access.cfc – generic DB class which is created to guarantee access to DB storage and data extraction by query execution.

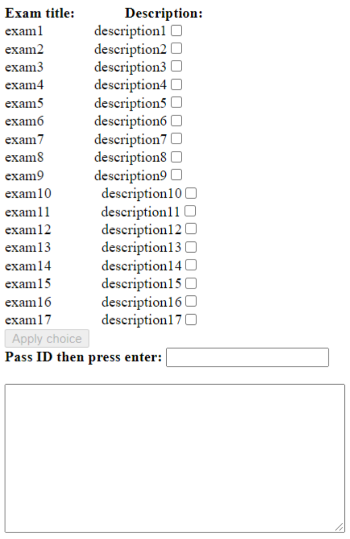
exams.cfc – extender class of previous. It is used for exam data extraction from DB storage by specific query execution.

testing_centers.cfc – It is used for testing centers data extraction from DB. Class structure is the same of the previous class, but query execution pattern is different.

Scheduler.cfc – class – wrapper. It is providing a flexible access to the functions of the previous classes.

Application workflow

This implementation of task assumes that users are already registered in the DB as well as exams and testing centers, the MySQL server which contains the DB is running as well as Lucee server. Once loaded (*default Lucee*) URL user is reached web-form which presented in Fig. 2.

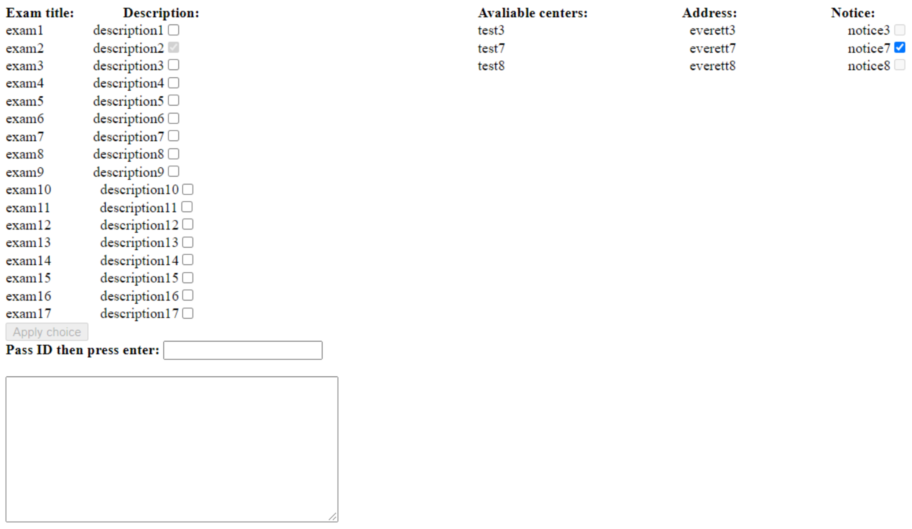


Exam title:	Description:
exam1	description1 ☐
exam2	description2 ☐
exam3	description3 ☐
exam4	description4 ☐
exam5	description5 ☐
exam6	description6 ☐
exam7	description7 ☐
exam8	description8 ☐
exam9	description9 ☐
exam10	description10 ☐
exam11	description11 ☐
exam12	description12 ☐
exam13	description13 ☐
exam14	description14 ☐
exam15	description15 ☐
exam16	description16 ☐
exam17	description17 ☐

Pass ID then press enter:

Fig. 2 – Initial page state

User should make a choice for any Exam using by checkboxes. *Note: The app. provides a possibility for making only one choice per time, so if user wants to apply for multiple Exams or for multiple Testing Centers it requires N times applying. The app. guarantees non-duplicating behavior i.e. if user applied for one Exam multiple times the app. guarantees one submission.* The second state is showed in the Fig. 3.



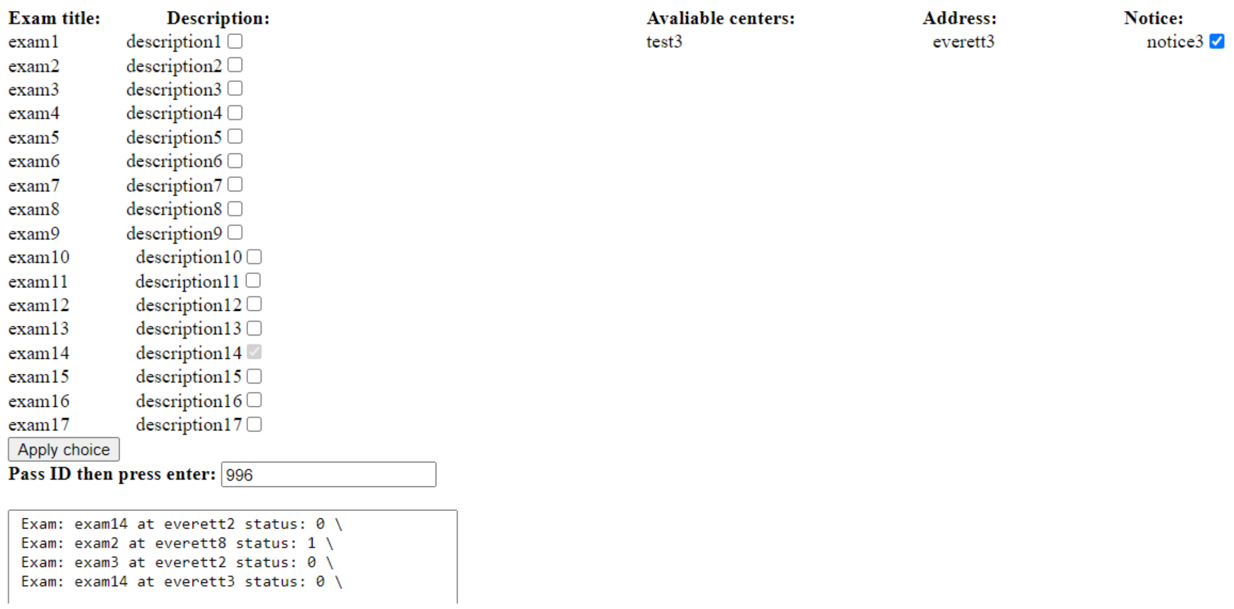
Exam title:	Description:
exam1	description1 ☐
exam2	description2 ✓
exam3	description3 ☐
exam4	description4 ☐
exam5	description5 ☐
exam6	description6 ☐
exam7	description7 ☐
exam8	description8 ☐
exam9	description9 ☐
exam10	description10 ☐
exam11	description11 ☐
exam12	description12 ☐
exam13	description13 ☐
exam14	description14 ☐
exam15	description15 ☐
exam16	description16 ☐
exam17	description17 ☐

Pass ID then press enter:

Avaliable centers:	Address:	Notice:
test3	everett3	notice3 ☐
test7	everett7	notice7 ✓
test8	everett8	notice8 ☐

Fig. 3 – Second page state with picked exam-testing center pair

At initial page state (Fig. 2) user press any of the checkboxes for pick Exam, then app. is processed the second page state when user should pick any of other checkboxes i.e. pick available testing center. *Note: user can pick only one testing center when it's done – other choices will be lock.* Further user should print personal ID to the input field which provided below than Exams list and press Enter on the keyboard for further processing. Then button 'Apply choice' will be unlocked. Once pressing the 'Apply choice' button user submits picked exam-testing center pair to further processing. The list of users Exams (user history) is printed in the text area below than ID text field (Fig. 4).



Exam title:	**Description:**
exam1	description1 ☐
exam2	description2 ☐
exam3	description3 ☐
exam4	description4 ☐
exam5	description5 ☐
exam6	description6 ☐
exam7	description7 ☐
exam8	description8 ☐
exam9	description9 ☐
exam10	description10 ☐
exam11	description11 ☐
exam12	description12 ☐
exam13	description13 ☐
exam14	description14 ✓
exam15	description15 ☐
exam16	description16 ☐
exam17	description17 ☐

Avaliable centers:	**Address:**	**Notice:**
test3	everett3	notice3 ✓

Pass ID then press enter: 996

```
Exam: exam14 at everett2 status: 0 \
Exam: exam2 at everett8 status: 1 \
Exam: exam3 at everett2 status: 0 \
Exam: exam14 at everett3 status: 0 \
```

Fig. 4 – Third and final page state with processed result of apply candidate to exam

Note1: The app. is providing limited feature. Users could apply only for available exams, but not reapply or cancel the choice. Extra functional could be simply added but not designed yet.

Note2: The data in DB is not cross-verify (hand-made input for test), so that is why in the Fig. 4 in the final history user already applied for exam14 at everett2, but it is not possible if user using app. functionality. It could be simple solved to start use app. with empty 'exam_associated' table.