Group 4:
Josh McManus
Maura Peterson
Rhett Harrison
Muhammad Fateen
Addison Corey
SER 322 - del Blanco
5 February 2023

Deliverable 3

*** The quotes - “ “ - implies the program will pose a query to the user and wait for an input response. ***

**Main Menu**

1. Course
2. Student
3. Garage
4. Staff
5. Infrastructure

**Course Sub Menu**

1.1. Create Course
1.2. Edit Course
1.3. Delete Course
1.4. Course (Report)
1.5. Weekly Schedule of Courses (Report)
1.6. Create Course Type
1.7. Edit Course Type
1.8. View Course Type
1.9. Delete Course Type

**Query for Menu 1.1 - Create Course**

UX will ask for course information and call DB with this query:
Parameter 1: course_name: varchar(30)
Parameter 2: course_type: int
Parameter 3: capacity: int
Parameter 4: course_description: varchar(150)
Parameter 5: cost: int
INSERT INTO course (course_name, course_type, capacity, course_description, cost) VALUES (‘?’, ?, ?, ‘?’, ?);

**Query for Menu 1.2 - Edit Course**

UX will ask for course id and information to change and call DB with this query:
Parameter 1: course_id: int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …
UPDATE course SET ?=? WHERE course_id=?;

**Query for Menu 1.3 - Delete Course**

UX will ask for course id and call DB with this query:

Parameter 1: course_id: int

DELETE FROM course WHERE course_id=?;

**Query for Menu 1.4 - Course Report**

UX will ask for course id and call DB with this query returning a list of student names and IDs enrolled in the course and whether they have paid or not:

Parameter 1: course_id: int

```
SELECT person.full_name,student.student_id,course_enrollment.paid
FROM course_enrollment,student,person
WHERE course_enrollment.student_id=student.student_id
AND student.person_id=person.person_id
AND course_enrollment.course_id=?;
```

**Query for Menu 1.5 - Weekly Schedule of Courses (Report)**

UX will ask for the start date of the week. We will calculate the date seven days later and call the DB with this query:

Parameter 1: week_start_date: Date

Parameter 2: week_start_date: Date

```
SELECT
course.course_id,course.course_name,course.course_type,course.capacity,course.course_description,co
urse.cost,course_schedule.course_date
FROM course,course_schedule
WHERE course.course_id=course_schedule.course_id
AND course_schedule.course_date >= ?
AND course_schedule.course_date < ?
ORDER BY course_schedule.course_date;
```

**Query for Menu 1.6 - Create Course Type**

UX will ask for course type information and call DB with this query:

Parameter 1: course_type_value: varchar(30)

INSERT INTO course_type (course_type_value) VALUES (‘?’);

**Query for Menu 1.7 - Edit Course Type**

UX will ask for course type id and information to change and call DB with this query:

Parameter 1: course_type_id: int

Parameter 2: Value to change

Parameter 3: New value

UPDATE course_type SET ?=? WHERE course_type_id=?;

**Query for Menu 1.8 - View Course Type**

UX will ask for course type id and call DB with this query:

Parameter 1: course_type_id: int

SELECT * FROM course_type WHERE course_type_id=?;

**Query for Menu 1.9 - Delete Course Type**

UX will ask for course type id and call DB with this query:

Parameter 1: course_type_id: int

DELETE FROM course_type WHERE course_type_id=?;

**Student Sub Menu**

2.1. Manage Student
2.2. Student Enrollment

**Manage Student Sub Menu**

2.1.1. Create Student
2.1.2. View Students
2.1.3. Edit Student
2.1.4. Delete Student
2.1.5. Student (Report)

**Query for Menu 2.1.1 - Create Student**

UX will ask for student information and call DB with this query:
Parameter 1: full_name: varchar(30)
Parameter 2: address: varchar(50)
Parameter 3: date_birth: date
Parameter 4: phone: varchar(10)

```
INSERT INTO person (full_name, address, date_birth, phone) VALUES ('?', '?', ?, '?');
INSERT INTO student (person_id) VALUES (
SELECT person_id
FROM person
WHERE full_name='?'
AND address='?'
AND date_birth=?
AND phone='?');
```

**Query for Menu 2.1.2 - View Students**

UX will call DB with this query:

```
SELECT student.student_id,person.full_name
FROM student,person
WHERE student.person_id=person.person_id;
```

**Query for Menu 2.1.3 - Edit Student**

UX will ask for student id and information to change and call DB with this query:
Parameter 1: student_id:  int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …

```
UPDATE person
SET ?=?
FROM person,student
WHERE person.person_id=student.person_id
AND student.student_id=?;
```

**Query for Menu 2.1.4 - Delete Student**

UX will ask for student id and call DB with this query:
Parameter 1: student_id:  int

DELETE FROM person
WHERE person_id=(
SELECT person_id
FROM student
WHERE student_id=?);
DELETE FROM course_enrollment
WHERE student_id=?;
DELETE FROM student
WHERE student_id=?;

**Query for Menu 2.1.5 - Student Report**

UX will ask for student id and call DB with this query:

Parameter 1: student_id: int

SELECT course_enrollment.course_id, course.course_name, course_enrollment.paid FROM course_enrollment,course WHERE course_enrollment.course_id=course.course_id AND course_enrollment.student_id=?;

**Student Enrollment Sub Menu**

2.2.1. Enroll Student
2.2.2. Unenroll Student

**Query for Menu 2.2.1 - Enroll Student**

UX will ask for student id and course id and call DB with this query:

Parameter 1: course_id: int
Parameter 2: student_id: int

INSERT INTO course_enrollment (course_id, student_id) VALUES (?, ?);

**Query for Menu 2.2.2 - Unenroll Student**

UX will ask for course enrollment id and call DB with this query:

Parameter 1: course_enrollment_id: int

DELETE FROM course_enrollment WHERE course_enrollment_id=?;

**Query for Menu 2.2.3 - View Student Enrollment**

UX will ask for course enrollment id and call DB with this query:

Parameter 1: course_enrollment_id: int

SELECT * FROM course_enrollment WHERE course_enrollment_id=?;

**Query for Menu 2.2.4 - Edit Student Enrollment**

UX will ask for course enrollment id, information to update, and call DB with this query:

Parameter 1: course_enrollment_id: int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …

UPDATE course_enrollment SET ?=? WHERE course_enrollment_id=?;

**Garage Sub Menu**

3.1. Create Bike
3.2. View Bikes
3.3. Edit Bike

3.4. Delete Bike
3.5. Assign Bike
3.6. Unassign Bike
3.7. Bike Problem Report
3.8. Bike Assignment Report
3.9. Create Bike Problem
3.10. View Bike Problem
3.11. Edit Bike Problem
3.12. Delete Bike Problem
3.13. Create Bike Type
3.14. View Bike Type
3.15 Edit Bike Type
3.16 Delete Bike Type

**Query for Menu 3.1 - Create Bike**

UX will ask for bike information and call DB with this query:

Parameter 1: brand: varchar(30)
Parameter 2: type: int
Parameter 3: license_plate: varchar(30)
Parameter 4: vin: varchar(30)
Parameter 5: broken: tinyint
Parameter 6: cc: int

INSERT INTO bike (brand, type, license_plate, vin, broken, cc) VALUES ('?', ?, '?', '?', ?, ?);

**Query for Menu 3.2 - View Bikes**

UX will call DB with this query:
SELECT bike_id, brand, cc, broken FROM bike;

**Query for Menu 3.3 - Edit Bike**

UX will ask for bike id and information to change and call DB with this query:

Parameter 1: bike_id: int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …

UPDATE bike SET ?=? WHERE bike_id=?;

**Query for Menu 3.4 - Delete Bike**

UX will ask for bike id call DB with this query:

Parameter 1: bike_id: int

DELETE FROM bike_assignment WHERE bike_id=?; // Delete from bike assignment
DELETE FROM bike WHERE bike_id=?;

**Query for Menu 3.5 - Assign Bike**

UX will ask for bike id and course schedule id and call DB with this query:

Parameter 1: bike_id: int
Parameter 2: course_schedule_id: int

INSERT INTO bike_assignment (course_schedule_id, bike_id) VALUES (?, ?);

**Query for Menu 3.6 - Unassign Bike**

UX will ask for bike_assignment_id and call DB with this query:
Parameter 1: bike_assignemnt_id: int
DELETE FROM bike_assignment WHERE bike_assignment_id=?;

**Query for Menu 3.7 - Bike Problem Report**

UX will ask for bike id and call DB with this query:
Parameter 1: bike_id: int
SELECT problem_date,repair_date,description,cost FROM problem WHERE bike_id=?;

**Query for Menu 3.8 - Bike Assignment Report**

UX will ask for bike id and call DB with this query:
Parameter 1: bike_id: int
SELECT course_schedule.course_date,course.course_name FROM
bike_assignment,course_schedule,course WHERE
bike_assignment.course_schedule_id=course_schedule.course_schedule_id AND
course_schedule.course_id=course.course_id AND bike_assignment.bike_id=?;

**Query for Menu 3.9 - Create Bike Problem**

UX will ask for bike id and problem information and call DB with this query:
Parameter 1: problem_date: Date
Parameter 2: bike_id: int
Parameter 3: repair_date: Date
Parameter 4: description: varchar(150)
Parameter 5: cost: int
INSERT INTO problem (problem_date, bike_id, repair_date, description, cost) VALUES (?, ?, ?, '?', ?);

**Query for Menu 3.10 - View Bike Problems**

UX will call DB with this query - will return all problems for all bikes:
SELECT * FROM problem;

**Query for Menu 3.11 - Edit Bike Problem**

UX will ask for problem id and information to change and call DB with this query:
Parameter 1: problem_id: int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …
UPDATE problem SET ?=? WHERE problem_id=?;

**Query for Menu 3.12 - Delete Bike Problem**

UX will ask for problem id call DB with this query:
Parameter 1: problem_id: int
DELETE FROM problem WHERE problem_id=?;

**Query for Menu 3.13 - Create Bike Type**

UX will ask for bike type information and call DB with this query:
Parameter 1: bike_type_value: varchar(10)
INSERT INTO bike_type VALUES (?);

**Query for Menu 3.14 - View Bike Types**

UX will call DB with this query:

SELECT * FROM bike_type;

**Query for Menu 3.15 - Edit Bike Type**

UX will ask for bike type id and information to change and call DB with this query:

Parameter 1: bike_type_id: varchar(10)

Parameter 2: Value to change

Parameter 3: New value

Parameter X: …

UPDATE bike_type SET ?=? WHERE bike_type_id=?;

**Query for Menu 3.16 - Delete Bike Type**

UX will ask for bike type id call DB with this query:

Parameter 1: bike_type_id: varchar(10)

DELETE FROM bike_type WHERE bike_type_id=?;

**Staff Sub Menu**

4.1. Coach

**Coach Sub Menu**

4.1.1. Create Coach

4.1.2. View Coaches

4.1.3. Edit Coach

4.1.4. Delete Coach

4.1.5. Assign Coach

4.1.6. Unassign Coach

4.1.7. View Coach Schedule

**Query for Menu 4.1.1 - Create Coach**

UX will ask for coach information and call DB with this query:

Parameter 1: full_name: varchar(30)

Parameter 2: address: varchar(50)

Parameter 3: date_birth: date

Parameter 4: phone: varchar(10)

Parameter 5: classroom_certified: tinyint

Parameter 6: dirtbike_certified: tinyint

Parameter 7: streetbike_certified: tinyint

INSERT INTO person (full_name, address, date_birth, phone) VALUES ('?', '?', ?, '?');

INSERT INTO coach (person_id, classroom_certified, dirtbike_certified, streetbike_certified)

VALUES ((SELECT person_id FROM person WHERE full_name='?' AND address='?' AND date_birth=? AND phone='?'), ?, ?, ?);

**Query for Menu 4.1.2 - View Coaches**

UX will call DB with this query:

SELECT coach.coach_id,person.full_name

FROM coach,person

WHERE coach.person_id=person.person_id;

**Query for Menu 4.1.3 - Edit Coach**

UX will ask for coach id and information to change and call DB with this query:

Parameter 1: coach_id:  int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …

```
UPDATE person
SET person.address='?'
WHERE person.person_id=(
SELECT DISTINCT person_id
FROM coach
WHERE coach_id=?);

UPDATE coach
SET coach.dirtbike_certified=?
WHERE coach_id=?;
```

**Query for Menu 4.1.4 - Delete Coach**

UX will ask for coach id and call DB with this query:

Parameter 1: coach_id:  int

```
DELETE FROM person
WHERE person_id=(
SELECT person_id
FROM coach
WHERE coach_id=?);
DELETE FROM coach_assignment
WHERE coach_id=?;
DELETE FROM coach
WHERE coach_id=?;
```

**Query for Menu 4.1.5 - Assign Coach**

UX will ask for coach id and role and course schedule id and call DB with this query:

Parameter 1: course_schedule_id: int
Parameter 2: coach_id:  int
Parameter 3: assigned_role: varchar(30)

```
INSERT INTO coach_assignment (course_schedule_id, coach_id, assigned_role)
VALUES (?, ?, '?');
```

**Query for Menu 4.1.6 - Unassign Coach**

UX will ask for the coach id and course schedule id and call DB with this query:

Parameter 1: coach_id:  int
Parameter 3: course_schedule_id: int

```
DELETE FROM coach_assignment
WHERE coach_id=?
AND course_schedule_id=?;
```

**View Coach Schedule Sub Menu**

4.1.7.1. Specific Coach Schedule
4.1.7.2. Availability Schedule

**Query for Menu 4.1.7.1 - Coach Weekly Schedule**

UX will ask for the start date of the week and the coach id. We will calculate the date seven days later and call the DB with this query:

Parameter 1: week_start_date: Date
Parameter 2: week_start_date: Date
Parameter 3: coach_id: int

```
SELECT DISTINCT
course.course_id,course.course_name,course_schedule.course_date,time_type.time_type_value
FROM course,course_schedule,time_type,coach_assignment
WHERE coach_assignment.coach_id=?
AND course.course_id=course_schedule.course_id
AND course_schedule.time_type_id=time_type.time_type_id
AND course_schedule.course_date >= '?'
AND course_schedule.course_date < '?'
ORDER BY course_schedule.course_date;
```

**Query for Menu 4.1.7.2 - Coach Availability Schedule**

UX will ask for the date and time type and call the DB with this query:

Parameter 1: date: Date
Parameter 2: time_type: ‘AM range’/’PM range’

```
SELECT * FROM coach c WHERE c.coach_id NOT IN (SELECT DISTINCT c1.coach_id
FROM coach c1, coach_assignment, course_schedule, time_type
WHERE c1.coach_id = coach_assignment.coach_id
AND coach_assignment.course_schedule_id = course_schedule.course_schedule_id
AND course_schedule.time_type_id = time_type.time_type_id
AND course_schedule.course_date = '?' AND time_type.time_type_value= '?');
```

**Infrastructure Sub Menu**

5.1. Range
5.2. Classroom

**Range Sub Menu**

5.1.1. Create Range
5.1.2. View Ranges
5.1.3. Edit Range
5.1.4. Delete Range
5.1.5. Assign Range
5.1.6. Unassign Range
5.1.7. View Range Schedule
5.1.8. Create Range Type
5.1.9. View Range Types
5.1.10. Edit Range Types
5.1.11. Delete Range Types

**Query for Menu 5.1.1 - Create Bike Range**

UX will ask for bike range information and call DB with this query:
Parameter 1: range_type: int
Parameter 2: capacity: int
INSERT INTO bike_range (range_type, capacity) VALUES (?, ?);

**Query for Menu 5.1.2 - View Bike Ranges**

UX will call DB with this query:
SELECT * FROM bike_range;

**Query for Menu 5.1.3 - Edit Bike Range**

UX will ask for range id and information to change and call DB with this query:
Parameter 1: range_id: int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …
UPDATE bike_range SET ?=? WHERE range_id=?;

**Query for Menu 5.1.4 - Delete Bike Range**

UX will ask for range id call DB with this query:
Parameter 1: range_id: int
DELETE FROM bike_range WHERE range_id=?;

**Query for Menu 5.1.5 - Assign Bike Range**

UX will ask for range id and course schedule id and call DB with this query:
Parameter 1: range_id: int
Parameter 2: course_schedule_id: int
INSERT INTO range_assignment (range_id, course_schedule_id) VALUES (?, ?);

**Query for Menu 5.1.6 - Unassign Bike Range**

UX will ask for the range id and course schedule id and call DB with this query:
Parameter 1: range_id: int
Parameter 3: course_schedule_id: int
DELETE FROM range_assignment WHERE range_id=? AND course_schedule_id=?;

**View Range Schedule Sub Menu**

5.1.7.1. Weekly Schedule of Range (Report)
5.1.7.2. Availability

**Query for Menu 5.1.7.1 - Weekly Schedule of Range (Report)**

UX will ask for the start date of the week and the range id. We will calculate the date seven days later and call the DB with this query:
Parameter 1: week_start_date: Date
Parameter 2: week_start_date: Date
Parameter 3: range_id: int

```
SELECT range_assignment.range_id,
course.course_id,course.course_name,course_schedule.course_date
FROM course,course_schedule,range_assignment
where course.course_id = course_schedule.course_id
```

AND course_schedule.course_schedule_id = range_assignment.course_schedule_id
AND course_schedule.course_date >= '?'
AND course_schedule.course_date < '?'
ORDER BY course_schedule.course_date;

**Query for Menu 5.1.7.2 - Range Availability**

UX will ask for the date and call the DB with this query:

Parameter 1: date: Date

```
SELECT  range_id
FROM bike_range
WHERE bike_range.range_id NOT IN
(SELECT ra.range_id
FROM range_assignment ra , course_schedule cs
WHERE cs.course_date = '?'
AND cs.course_schedule_id = ra.course_schedule_id
GROUP BY range_id
HAVING count(*) >= 2);
```

**Query for Menu 5.1.8 - Create Range Type**

UX will ask for range type information and call DB with this query:

Parameter 1: range_type_value: varchar(30)

```
INSERT INTO range_type (range_type_value) VALUES (‘?’);
```

**Query for Menu 5.1.9 - View Range Types**

UX will call DB with this query:

```
SELECT * FROM range_types;
```

**Query for Menu 5.1.10 - Edit Range Type**

UX will ask for range type id and information to change and call DB with this query:

Parameter 1: range_type_id:  int
Parameter 2: Value to change
Parameter 3: New value
Parameter X: …

```
UPDATE range_type SET ?=? WHERE range_type_id=?;
```

**Query for Menu 5.1.11 - Delete Range Type**

UX will ask for range type id call DB with this query:

Parameter 1: range_type_id:  int

```
DELETE FROM range_type WHERE range_type_id=?;
```

**Classroom Sub Menu**

5.2.1. Create Classroom
5.2.2. View Classrooms
5.2.3. Edit Classroom
5.2.4. Delete Classroom
5.2.5. Assign Classroom
5.2.6. Unassign Classroom

5.2.7. View Classroom Schedule

**Query for Menu 5.2.1 - Create Classroom**

UX will ask for classroom information and call DB with this query:

Parameter 1: capacity: int

INSERT INTO classroom (capacity) VALUES (?);

**Query for Menu 5.2.2 - View Classrooms**

UX will call DB with this query:

SELECT * FROM classroom;

**Query for Menu 5.2.3 - Edit Classroom**

UX will ask for classroom id and information to change and call DB with this query:

Parameter 1: classroom_id: int

Parameter 2: Value to change

Parameter 3: New value

Parameter X: …

UPDATE classroom SET ?=? WHERE classroom_id=?;

**Query for Menu 5.2.4 - Delete Classroom**

UX will ask for classroom id call DB with this query:

Parameter 1: classroom_id: int

DELETE FROM classroom WHERE classroom_id=?;

**Query for Menu 5.2.5 - Assign Classroom**

UX will ask for classroom id and course schedule id and call DB with this query:

Parameter 1: classroom_id: int

Parameter 2: course_schedule_id: int

UPDATE course_schedule SET classroom_id=? WHERE course_schedule_id=?;

**Query for Menu 5.2.6 - Unassign Classroom**

UX will ask for the classroom id to replace the removed classroom with and course schedule id and call DB with this query:

Parameter 1: classroom_id: int

Parameter 3: course_schedule_id: int

UPDATE course_schedule SET classroom_id=? WHERE course_schedule_id=?;

**View Classroom Schedule Sub Menu**

5.2.7.1. Specific Classroom Schedule

5.2.7.2. Availability

**Query for Menu 5.2.7.1 - Classroom Schedule**

UX will ask for the start date of the week and the classroom id. We will calculate the date seven days later and call the DB with this query:

Parameter 1: week_start_date: Date

Parameter 2: week_start_date: Date

Parameter 3: classroom_id: int

```
SELECT
course_schedule.classroom_id,course.course_name,course_schedule.course_date,time
_type.time_type_value
FROM course,course_schedule,time_type
WHERE course.course_id=course_schedule.course_id
AND course_schedule.time_type_id=time_type.time_type_id
AND course_schedule.course_date >= ?
AND course_schedule.course_date < ?
and course_schedule.classroom_id in (select classroom_id from classroom)
ORDER BY course_schedule.course_date;
```

**Query for Menu 5.2.7.2 - Classroom Availability**

UX will ask for the date to check:

Parameter 1: date: Date

```
SELECT classroom_id
FROM classroom
WHERE classroom.classroom_id not in
(SELECT classroom_id
FROM course_schedule cs
WHERE cs.course_date = '?'
AND classroom_id in
    (SELECT classroom_id
    FROM classroom)
GROUP BY classroom_id
HAVING count(*) >= 2);
```

**Query for Menu 5.2.8 - Create Course Schedule**

UX will ask for course schedule information and call DB with this query:

Parameter 1: classroom_id: int

Parameter 2: course_date: Date

Parameter 3: course_id: int

Parameter 4: time_type_id: int

```
INSERT INTO course_schedule (classroom_id, course_date, course_id, time_type_id)
VALUES (?, ?, ?, ?);
```

**Query for Menu 5.2.9 - View Course Schedule**

UX will call DB with this query:

```
SELECT * FROM course_schedule;
```

**Query for Menu 5.2.10 - Edit Course Schedule**

UX will ask for course schedule id and information to change and call DB with this query:

Parameter 1: course_schedule_id:  int

Parameter 2: Value to change

Parameter 3: New value

Parameter X: …

```
UPDATE course_schedule SET ?=? WHERE course_schedule_id=?;
```

**Query for Menu 5.2.11 - Delete Course Schedule**

UX will ask for course schedule id call DB with this query:

Parameter 1: course_schedule_id:  int

DELETE FROM course_schedule WHERE course_schedule_id=?;

**Query for Menu 5.2.12 - Create Time Type**

UX will ask for time type information and call DB with this query:

Parameter 1: time_type_value: varchar(30)

INSERT INTO time_type (time_type_value) VALUES (?);

**Query for Menu 5.2.13 - View Time Type**

UX will call DB with this query:

SELECT * FROM time_type;

**Query for Menu 5.2.14 - Edit Time Type**

UX will ask for time type id and information to change and call DB with this query:

Parameter 1: time_type_value:  int

Parameter 2: Value to change

Parameter 3: New value

Parameter X: …

UPDATE time_type SET ?=? WHERE time_type_id=?;

**Query for Menu 5.2.15 - Delete Time Type**

UX will ask for time type id call DB with this query:

Parameter 1: time_type_id:  int

DELETE FROM time_type WHERE time_type_id=?;